AFIT/GSO/ENG/92D-05

Correlation Based Target Location and Identification

THESIS

Richard Alan Wenzel
Captain, USAF

AFIT/GSO/ENG/92D-05

Approved for public release; distribution unlimited

AFIT/GSO/ENG/92D-05

Correlation Based
Target Location and Identification

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science (Space Operations)

Richard Alan Wenzel, B.S.
Captain, USAF

December, 1992

Approved for public release; distribution unlimited

Preface

The purpose of this study was to determine the viability of correlation as an autonomous target locator. A correlator was built which utilized the routines inherent in the KHOROS software package developed by the University of New Mexico. A set of 180 visible light, terrain board images were correlated before and after pre-processing and the results compared against the known target location. A set of rules, both quantitative and heuristic, were developed as the basis for a rule-based expert system.

I wish to express my gratitude to my thesis advisor, Maj Steve Rogers, for his wisdom and patience as I discovered academic areas I had never experienced before. I would like to thank Capt Dennis Ruck, who provided valuable assistant in writing the computer code necessary for this effort. I also would like to thank Maj John Borsi, who asked the toughest question of all: "So what?".

I would also like to thank my fellow members of GSO class 92-D for all of their help and friendship. You put new meaning to the phrase "cooperate and graduate".

Most of all, I want to thank my beautiful wife, Terri, and my wonderful daughter, Bailey Michelle. Terri, any attempt to thank you would be trite at best. You kept our home a safe refuge from the AFIT monster, and allowed me to ignore the ever present dishes and cat litter more times than I can count. Thanks for taking care of me and allowing me to do what I needed to do, and for being sane when the rest of the world was not. Bailey, you came into this world here, and you know no life other than AFIT. I promise to be around more. Thank you for allowing me to rock you to sleep, feed you, and change your diaper at the times when I needed to the most. This degree belongs to you both as much as it does to me.

Richard Alan Wenzel

Table of Contents

List of Figures

Correlation Based
Target Location and Identification

I. Introduction

With the current Department of Defense drawdown, the military of the future will have to provide for the nation's security with fewer people. One way to minimize the effects of this drawdown is by developing smart weapons that can autonomously sense, identify, track, and destroy enemy forces. These weapons would allow fewer people to operate more equipment, thus increasing the offensive capabilities of a smaller military force.

Development of weapons that can autonomously find and destroy enemy forces is an extremely complex process which relies on research in many areas of pattern recognition, sensor design, and computer science. For a weapon to locate and destroy a target, it must have 1) sensors capable of locating possible targets, 2) an automated target recognizer (ATR) to discriminate targets from non-targets or background clutter, and provide this information to the software to make the destroy or not destroy decision, and 3) a guidance system to lock-on and track the target until destruction. The first and third steps in this process are already within the reach of current technology. However, it is the second step that remains the unsolved problem.

The Air Force has spent over 25 years researching and developing hardware and software in the attempt to design an ATR, for battlefield situations. However, due to the complicated nature of the problem, a true ATR system does not exist today.

1.1 Background

There are many systems today that provide a limited solution to the object recognition problem. Some supermarkets use a laser beam to read the universal product code placed on consumer goods, ensuring the correct price is charged to the customer. Similarly, some secure Air Force bases use scanners that read the unique pattern of blood vessels found in the retina for personnel identification. While these examples work well, they exist in a controlled environment. The object's orientation, position, and motion are constrained to within certain parameters to guarantee quick processing. In fact, it behooves the individuals involved to ensure that the targets are presented to the system correctly. Otherwise, the checkout clerk at the supermarket will incur the wrath of the customer for being too slow, and the person seeking entry to the base will be temporarily detained in the retina scanning booth until the security police release them.

Even with the many years of research in this area, the best target recognizer in existence remains not a machine, but something found in nature: the biological vision system. While many algorithms have been developed to simulate what the eye and brain do naturally, only limited successes have been reported. In fact, none of these successes can match what even a pigeon is capable of doing. As reported by Rogers (15:35), B. F. Skinner, a famous psychologist, trained pigeons to recognize whether people were present in an outdoor scene. According to Cerella (5:431), even though pigeons have relatively small brains, they can still sort slides into natural categories such as people, trees, and bodies of water, a capability that humans do easily. As such, pigeons could detect people in a scene approximately 76 out of 80 times. In fact, the pigeons' discrimination abilities were found to be good enough that they would be able to "provide all the guidance necessary, to the accuracy desired", to deliver munitions on target (15:36).

The performance of any ATR system is dependent on good sensor data. Assuming good sensor data is available to the ATR system, the task of automated target

recognition is classically divided into three steps. First, *segmentation* labels individual pixels and groups sets of pixels into classes, usually possible target or non-target. Second, *feature extraction* computes values which represent the distinguishing characteristics of the pixel groups found by segmentation. Finally, *classification* assigns a label, usually which type of target, to the pixel groups based on the set of features found during feature extraction. However, specific techniques for ATR may "blur" this three step process into a different set of steps.

A common technique used for target location and identification is correlation (1:66) (9:92). In correlation, a representative image of a target, called a *template*, is compared to the image of interest with the goal of finding the target in the image. The template is "overlaid" onto the image of interest at every possible position. For each new template position, the correlation between the template and the image is calculated. The correlation function is defined mathematically as (8:172):

$$R_{ts}(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} i(\alpha,\beta)t^*(\alpha-x,\beta-y)d\alpha d\beta \qquad (1)$$

where $i(\alpha,\beta)$ is the image being processed, $t(\alpha-x,\beta-y)$ is the template the image is correlated with, * represents the complex conjugation operation, and R_{ts} represents the resulting correlation. This correlation represents the degree of similarity between the template and the image for that particular pixel in the image.

In a perfect world, the highest correlation calculated, called the *correlation peak*, represents the position in the image where the best match between the template and the image occurred. However, in a non-perfect world, the actual target location may not be located at the peak, but at the correlation minimum, or at a secondary peak. Correlating an image with various templates will result in potential targets found for each template. The decision then becomes which template best matches the image.

While correlation is a useful tool, it does have some limitations. First, it is sensitive to scale and rotation changes (10:183). A template represents only a target of a particular size and rotational position. Should an image change orientation or size due to movement relative to the sensor, the correlation peak will degrade from the peak correlation value. If the target continues to move, the correlation peak may fade below the threshold and escape detection. An example of this can be found in research conducted by Casasent and Psaltis (4:1795). In their experiments, the signal-to-noise ratio decreased from 30 dB to 3 dB with only a two percent scale change and a 3.5 degree rotation change of the image being processed. The strength of the signal as compared to the noise decreased from 1000 times greater, to only two times greater.

One way to overcome these problems is by pre-processing the raw image data to increase the chance for a successful correlation. Another possible way to alleviate these problems is to use numerous templates that represent different scale and rotation variations for each possible target.

A second limitation is the number of computations needed for correlation. For an image that is 128 x 128 pixels in size, 16384 correlation values must be calculated, each representing one of the 16384 possible positions in the image. Given a template that is 32 x 32 pixels in size, a total of 1024 pixel-to-pixel calculations occur for *each* of the 16384 possible positions. In addition, a system will normally use numerous templates to represent the targets in various scales and orientations. Compounding this is the rate at which the image will be updated in the true target acquisition scenario. If the sensor stores a representation of the scene every second, the number of computations necessary quickly becomes prohibitive. However, there is a way to correlate an image with a template using a Fourier transformation (9:81). The frequency domain representation of the correlation function can be written as:

$$R_{ts}(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(\xi,\eta)T^*(\xi,\eta)exp[j2\pi(\xi x+\eta y)]d\xi d\eta \quad (2)$$

$$= \mathcal{F}^{-1}\{\mathcal{F}[i(x,y)]\mathcal{F}[t(x,y)]^*\} \quad (3)$$

$$= \mathcal{F}^{-1}[I(u,v)T^*(u,v)] \quad (4)$$

where ξ and η are the spatial frequency coordinates, $I(\xi,\eta)$ is the image in the frequency domain, and $T^*(\xi,\eta)$ is the complex conjugate of the frequency domain representation of the template, $\mathcal{F}$ is the Fourier transformation operation, and $\mathcal{F}^{-1}$ is the inverse Fourier transformation operation. Each image and template is transformed from the space domain to the Fourier domain, resulting in both a real and imaginary number for each pixel in both the template and the image. The template is further processed by conjugation, which simply reverses the sign on the imaginary part of each pixel. The image and template are then multiplied pixel by pixel and inverse Fourier transformed back to the space domain. The resulting image is the 2-D correlation plane as described above, but was found with far fewer computations for any reasonably sized image.

Another limitation of correlation is its sensitivity to background clutter. Where no distribution of background clutter is known *a priori*, the target of interest can be effectively hidden from the correlator. In fact, a pixel with a large intensity value can be sufficient to skew the correlation peak completely off the target (1:67). To overcome this limitation, a technique called *energy normalization* can be used to maximize the target energy or contrast with respect to the background clutter.

This research will focus on the problem of locating and identifying targets within a cluttered background using correlation via Fast Fourier Transforms and energy normalization. No assumption will be made as to the amount of background clutter resident in the image.

1.2 Problem Definition

A reliable target recognition system capable of locating and identifying objects in cluttered images does not exist today. This system must process sensor data, reliably identify targets, and provide appropriate information to the weapon guidance system. This effort will investigate correlation to determine if it is a viable technique for locating and identifying targets in visible light terrain board imagery.

1.3 Materials and Equipment

The images used for this effort are photographs of terrain boards. A terrain board is a physical scale model of possible hostile targets among various backgrounds and occlusions. It is made up of smaller scale versions of objects possibly found on a battlefield, such as bushes, trees, hills, valleys and boulders, as well as hostile tanks, trucks, and airplanes. These objects are arranged in various configurations to simulate battlefield topography and the enemy's use of topography for camouflage or cover during combat situations. The images are in the visible frequency domain, that is, they represent black and white images discernable by the human eye.

A total of 180 images were used for this project. All images are visible light terrain board images of M1 tanks, trucks, and airplanes with varying amounts of clutter, ambient light, and camera elevation. Specifically, the images used for this project consist of 84 M1 tank images with various aspect angles and amounts of background clutter (reference Figure 1), 18 airplane images showing airplanes from above (reference Figure 2), and 78 truck images, also showing the trucks from above (reference Figure 3).

These images are stored in the HIPS format which is basically a header ending with a period (.) followed by the binary gray scale representations of the pixels in the image. Each pixel is encoded with a brightness intensity on the scale from absolute black (0) to absolute white (255). The image consists of 16384 pixel values which correspond to an image size of 128 x 128 pixels. A more detailed description of the

Figure 1. Sample Tank Image

HIPS format can be found in Appendix B along with routines to perform various image format conversions found in Appendix C.

The hardware used for this research consist of the SUN SPARC 2 workstations and the Silicon Graphics workstations. The software used consists of both "off-the-shelf" and application specific programs. The "off-the-shelf" software was KHOROS, an integrated image and signal processing software environment developed by the faculty and students at the University of New Mexico(14). KHOROS is basically a large library of image and signal processing routines that can be used singly or in groups to perform such functions as transforms and data conversions. The application specific programs consist of C programs used for image pre-processing, file manipulations, and data conversions written specifically for this effort.

1.4 Scope and Limitations

This research concentrates only on the problem of target location and identification in cluttered images. Specifically, this effort will evaluate the effectiveness of

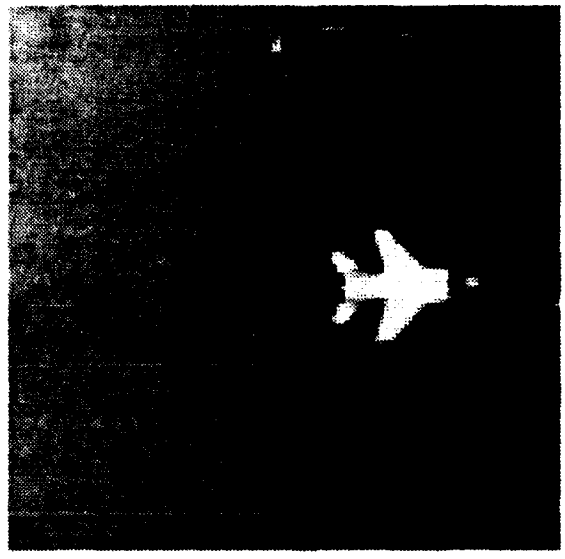


Figure 2. Sample Plane Image

correlation as a target locator and identifier in visible light imagery. The following limitations are imposed on this effort:

- Detecting the presence of a target will not be addressed. All images are assumed to have a target present. Previous AFIT research by Mayo (13), Troxel (20), and Walrond and Childress(21) have addressed target detection.
- The optical implementation and testing of the research methods will not be conducted. These are subjects worthy of their own research efforts.
- Multiple target recognition will not be addressed. Each image is assumed to have only one target present.

1.5 Thesis Overview

This thesis is organized into five chapters with seven appendices. The material in the following chapters is grouped logically by types of images processed: raw images and energy normalized images. The appendices contain all software developed

Figure 3. Sample Truck Image

in support of this effort as well as background information provided to help readers with less experience in correlation image processing.

This chapter is an introduction to the thesis topic, its scope, and materials and equipment used. Chapter II contains a literature review showing how similar problems have been studied in the past. Chapter III discusses the research methodology and tests conducted to verify the integrity of various software routines developed. Chapter IV reports the results of the research. Chapter V presents conclusions reached through this effort.

II. Literature Review

This chapter presents past research efforts that attempted to solve similar ATR problems.

2.1 Daugman Research

Daugman (7) cites research on the mammalian visual nervous system (retina, lateral geniculate, and primary visual cortex) as motivation for using Gabor filters to transform input images and achieve better segmentation results. He proves that 2-D (two-dimensional) Gabor filters can "capture critical neurobiological variables of a given neuron's orientation and spatial frequency preference, the tuning bandwidths for these variables, the receptive field dimensions and the relationships among all of these parameters". In fact, 97 percent of mammalian cortical simple cells have 2-D receptive field profiles which can be well fit by 2-D Gabor elementary functions.

Daugman used a three layered artificial neural network to find better Gabor filters. The network had fixed weights on the first and third layers based on the 2-D Gabor elementary functions and the second layer having adjustable weights. The input image was composed of various textures of anisotropically filtered white noise fields and was successfully segmented into its various texture regions.

2.2 Hazlett Research

Hazlett (11) investigated segmenting high resolution synthetic aperture radar (SAR) images using both Gabor filters and radial basis functions. Each 2048 x 2048 pixel SAR image was normalized via a fast Fourier transform to minimize the differences due to various image sources.

The initial set of Gabor filters were those suggested by Daugman. From this starting point, complete Gabor filter sets for various window sizes were generated.

The optimal set of filters were chosen using two methods. First, the power spectrum was computed and the two highest peaks were chosen. Second, specific windows of the input image with known content, such as trees, grass,and shadows, were processed and the three filters having the greatest magnitude coefficients were selected. The filters thus selected were then combined and the three filters with the greatest occurrences were selected as the final set.

Hazlett then used the Gabor filter coefficients as training vectors into a radial basis function neural network. The number of clusters used to build the neural network was varied so that an optimum number could be found. From this, he achieved an overall accuracy of 81 percent. However, there was some shortcomings to this process. The transition regions between trees and shadows were often misclassified as grassy which limits the overall accuracy of the process.

2.3 Ruck Research

Ruck (17) studied ways to segment an image using a multi- function laser radar sensor using both doppler and relative range images. He used two techniques to segment the different types of images, optimum thresholding, and the technique developed by C. Tong in his 1986 thesis *Target Segmentation and Image Enhancement through Multisensor Data Fusion* (19).

To segment the doppler images, he used three different threshold based approaches which all were ultimately binary optimum thresholding techniques. The optimum threshold approach involves the idea that there exist only two type of pixels in the image, in this case, target and nontarget. Any pixel with a value higher than the threshold value is considered a target pixel and any below the threshold is considered a nontarget pixel.

However, for segmentation of the relative range images, he used the algorithm developed by C. Tong. This algorithm assumed that the range gradient (the difference between maximum and minimum values of the range) will be relatively low for

targets while the range gradient will be relatively high for nontarget (background) images.

From the two segmentation techniques, Ruck was able to create a list of all the different regions found in the input images. He did this using two different classification decision rules:

- a statistical nearest neighbor approach
- a biologically-based neural network multilayer perceptron

Ruck's conclusion is that the multilayer perceptron performed is statistically equivalent to the nearest neighbor classifier.

2.4 Roggemann Research

Roggemann (16) studied Forward-Looking Infrared (FLIR) sensor images to develop a segmentation algorithm. He based his algorithm on the following observations:

- the targets usually have higher temperatures than the background
- the targets usually were differentially heated due to operation and sun warming
- the targets occupied a small fraction of the total pixels found in the image

With these assumptions, an optimum threshold approach was used. Since FLIR images are a measure of the apparent relative temperature of the pixels, and targets are considered hotter than the background, any pixel higher than a threshold value is considered a target pixel and any lower than that value is considered a background pixel. However, selecting a threshold value is a problem. With two types of regions resident in the image, most expect a bi-modal distribution when the pixel intensities are placed in a histogram and a good threshold value would be the mean of the two median values (9:354). In Roggemann's imagery, the target pixels are relatively rare

as compared to the background pixels. This resulted in the histogram representation of pixel values not showing the expected bi-modal distribution.

To overcome this problem, Roggemann developed an adaptive algorithm to choose the appropriate threshold value and thus segment the image. He then used a set of heuristic operations to reject various pixels that were characterized as targets but were deemed not targets. Results of this ATR system were found to be useful, but not excellent. Roggemann cites the need for development of better heuristic segmentation operations.

2.5 Smiley Research

Smiley (18) showed that cluttered SAR imagery could be segmented using parts of a multiresolution representation produced by dilations and translations of a wavelet transform. This process involved three steps:

- generating the multiresolution representation of the image
- extracting the features from that representation
- segment the multiresolution representation using a radial basis function artificial neural network into different homogeneous regions such as trees, fields, and shadows

Each image is transformed using an affine wavelet, which can be thought of as filtering the image through a progression of low-pass filters with decreasing bandwidths. This results in a set of representations of the original image, each with successively lower resolution. From here, an optimum level of representation is chosen. This chosen level has associated with it certain values which represent the frequency content of the image. These values are then used as features and fed into a radial basis function artificial neural network as a way to train the network. Images to test the network were similarly transformed and fed into the neural network.

Smiley's results show that the neural network could segment both natural regions as well as man-made objects from the surrounding region.

2.6 Law Research

Law (12) investigated object tracking using adaptive correlation techniques. Forward Looking Infrared Radar (FLIR) images of various targets were collected at one second intervals from a DC-3 aircraft as it flew from a range of roughly 10 kilometers until it was directly above the targets. These images were then digitized for processing using adaptive correlation. Each image processed by the adaptive correlator became the template for the subsequent image processing. In other words, the correlator "adapted" to the changing scene by updating the template each processing cycle. This prevented the correlation peak from fading as the target deviated from the scale and/or rotation represented in the image.

Law's results showed adaptive correlation to be a viable technique for tracking infrared targets if two conditions were met. First, the time difference between successive frames must be small, representing a high scan rate, and the processing cycle of the correlator must be as fast, or faster, than the image scan rate. Second, an image enhancement technique must be used (12:95-96).

2.7 Booz, Allen, & Hamilton Research

Previous work by Booz, Allen, & Hamilton for Wright Laboratories Advanced Systems Research Group (WL/AAAT-1) was successful in segmenting visible light, terrain board imagery (2). Their approach was to have an operator hand segment representative images to extract the target of interest for use as templates. These templates were then used to extract 11 x 11 sub-images, called *features*, which became the templates used for the correlation step. The relative spatial locations of the features in the original image were also noted for use in the performance (object location) phase. The features and spatial locations were then fed into the Artificial

Neural Vision Learning System (*ANVIL*) which used a neural network architecture for the target identification process.

This work was successful in that it proved to be robust to scale and perspective changes, provided that it was trained with imagery that had a variety of scales and perspectives represented. The amount of robustness was found to be ± 10 degrees in azimuth, ± 5 degrees in elevation, and ± 25 percent in scale(2:2).

2.8 Summary

Overall, there are many techniques that attempt to simulate what the mammalian brain does naturally. Each technique contains inherent strengths to be maximized and limitations to be overcome. Investigating all of them would require much more time than allowed for this effort. As such, only one technique will be studied: correlation. A description of the correlator and its implementation as well as the research methodology follow in the next chapter.

III. Research Methodology

The correlator used on the raw images for this effort was implemented by routines inherent in the KHOROS software and is similar to the correlator developed by Law (12). A description of the routines used is located in Appendix E.2.

Once the correlator was built, the next step was to verify the integrity of the correlator by a series of tests. A 128 x 128 pixel test image was created which is composed of a completely white square target (pixel values of 255.0) on a black (pixel values of 0.0) background. The test image also contains some white speckle and a 10 x 10 pixel region of high intensity pixels. The target has an upper left coordinate of (83,42) and has a width and height of 30 pixels. The known center of the target is at pixel location (98,57). The test image is shown in Figure 4.

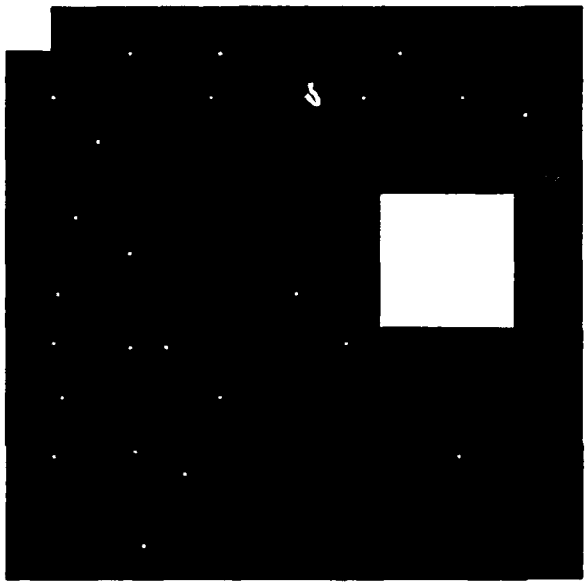


Figure 4. Correlator Test Image

The purpose of the tests was to verify the operation of the correlator and its sensitivity to noise and intensity variations. Results of the test are included in the next chapter.

The following steps compose the research methodology:

- Truth data - Each test image was hand-truthed. The smallest box that completely surrounds the target, without touching the target, was noted and its centroid calculated by the following equation:

$$x_{mid} = x_{min} + int(\frac{width}{2})$$

$$y_{mid} = y_{min} + int(\frac{height}{2})$$

where x_{mid} and y_{mid} represent the centroid of the target, x_{min} and y_{min} represent the "upper left" coordinate values used to truth the image using KHOROS (by KHOROS convention), width is the target width in pixels, and height is the target heights in pixels. These four values, x_{min}, y_{min}, *width*, and *height* completely define the box drawn around the target and form the basis for all correlator experiments. A list of the image names and the truthing data used for correlation is found in Appendix D.

- Image pre-processing - Each image was pre-processed by energy normalization and local energy normalization to enhance the effectiveness of correlation as a target recognizer.

- Pre-processed Image Correlation - These pre-processed images were then correlated with the known target inherent to that image. The resulting correlation peak and correlation minimum were noted. The images were then correlated with various templates of similar targets with the correlation peaks noted. Finally, the entire energy normalized image set was correlated across all templates and the peaks noted.

- Raw Image Correlation - Each test image was correlated with a template made from the target located in that image surrounded by minimum value pixels.

The resulting correlation peak, correlation minimum, and secondary peaks were noted as possible target locations.

- Comparison - The correlation peak and/or correlation minimum found from the various correlation steps were compared with the known target centroid and the distance between the two calculated by the following equation:

$$D = \sqrt{(x_{mid} - x_c)^2 + (y_{mid} - y_c)^2}$$

where x_c and y_c represent the position of the correlation peak or minimum, and D represents the distance in pixels.

- Heuristic target location - Any image that failed both the correlation peak and correlation minimum criteria had its 3-D correlation plane examined for a secondary peak. When this failed, the 2-D correlation plane was also examined in an attempt to identify a heuristic rule or rules for locating the target.

- Success/Failure decision - The distance D calculated was used to determine success or failure. Any distance greater than one half the diagonal of the box encircling the target was considered a failure. Also, any image failing the distance criteria was examined for success/failure using the heuristic rules developed. When the entire image set was correlated with the entire template set, the best correlation peak noted was also examined as to which type of target had been identified, with an incorrect target identification resulting in a failed correlation.

3.1 Summary

This chapter reported the correlator and research methodology used for this effort. The next chapter reports results from the correlator operations checks, followed by the actual correlation results and success rates.

IV. Findings

This chapter reports the results achieved in this effort. The first section reports the operational check of the correlator. The second section reports the sensitivity of the correlator to low variability pixel images. The final section reports the actual correlation results achieved on the set of images.

4.1 Correlator Operation Check

The first test checks for proper correlator operation. The target (reference Figure 5) was extracted and centered in the template, with a background of zero pixel values (Reference Figure 6). The test image was then correlated with the template and the correlation peak noted (Reference Figure 7 and Figure 8). The statistics from the test image correlation (reference Figure 9) show the correlation peak was found at pixel location (98,57) as expected. An additional check can be made with simple mathematics. Correlating a 30 x 30 square of 255.0 pixel values with a 30 x 30 square of 255.0 pixel values resulted in a correlation peak of $30^2 \times 255^2 = 5.85225x10^7$. This also matches the magnitude of the correlation peak found in Figure 9.

4.2 Correlator Sensitivity Check

The next test checked the sensitivity of the correlator. The test image is identical to the previous test image, except that the background pixels are set to 254.0 rather than 0.0. If the correlator is working correctly, then the correlation peak will also be found at pixel location (98,57) as noted above. Referencing Figure 10 and Figure 11, the correlation peak was found where it should be.

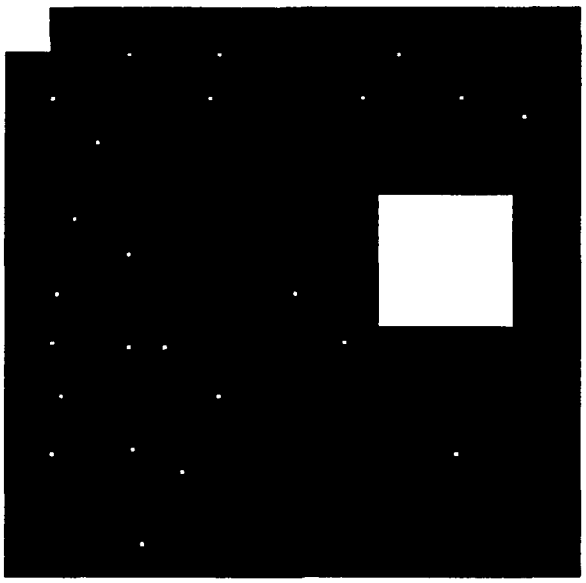


Figure 5. Correlator Test Image

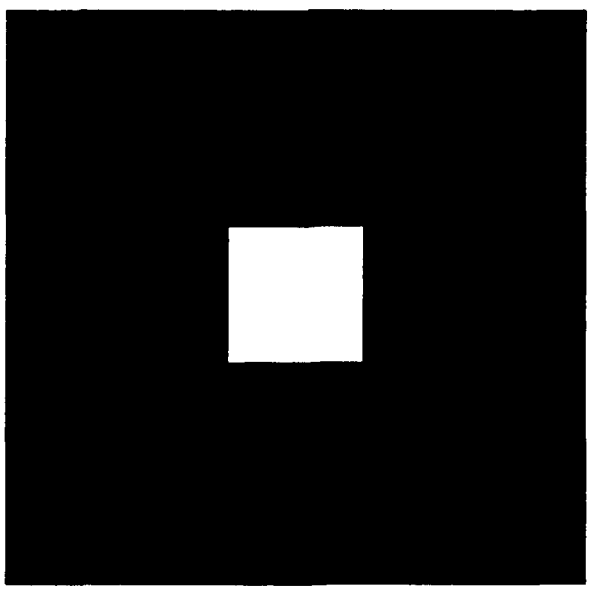


Figure 6. Correlator Test Image Template

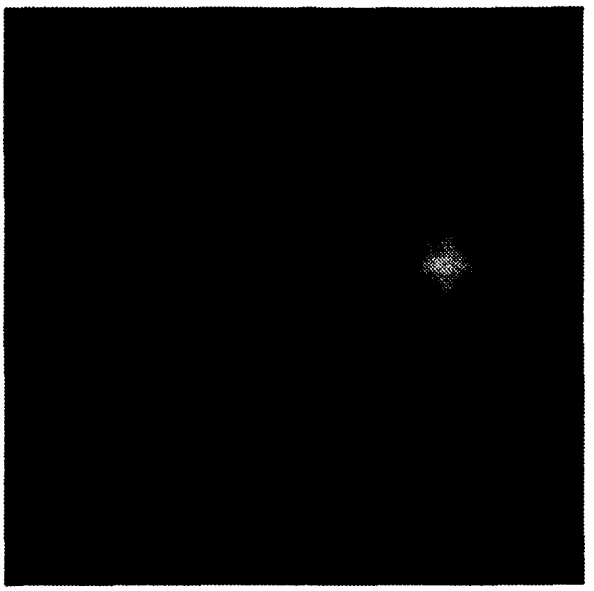


Figure 7. 2-D Test Image Correlation Plane

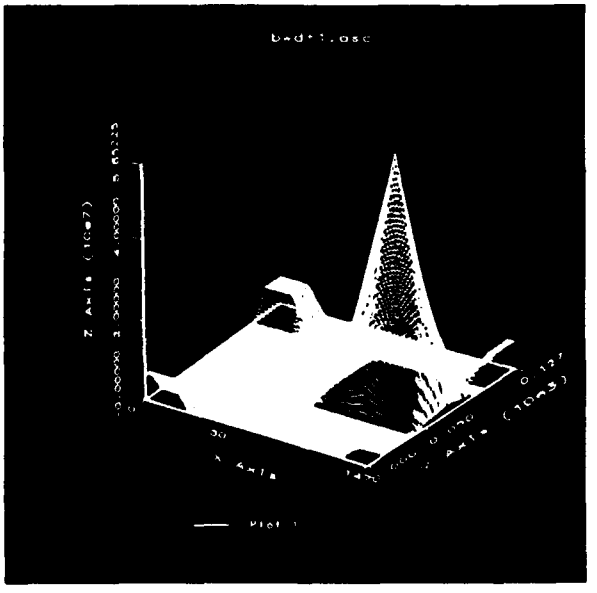


Figure 8. 3-D Test Image Correlation Plane

```
Image Statistics for File Name: /tmp/vctorKMAa17713

******* Statistics for Band #0 ******

  Mean: 3.65766e+06
  Variance: 7.23352e+13
  Std. Dev: 8.50501e+06
  RMS:  9.25793e+06
  Peaks:
    High: 5.85225e+07 at (X,Y) location: (98,57)
    Low : -4 at (X,Y) location: (43,60)
  Total integral under the image: 5.9927e+10
  Positive part of integral under the image: 5.9927e+10
  Negative part of integral under the image: -260.386
  Contributing points: 16384
  Number of positive points in image: 15914
  Number of negative points in image: 470
  Skewness:  3.13534
  Kurtosis:  10.2604
```

Figure 9. Test Image Correlation Statistics

4.3 Energy Normalization Check

The final test checks for proper energy normalization of the images. Each image was normalized using the *enorm.c* program found in Appendix D.8 and stored in a new file. This new file was then used as the input into the program *check_norm.c* found in Appendix D.11. This program simply read in the energy normalized file one pixel at a time, squared the pixel value, and then added this value to a running sum. When completed, the running sum should be equal to the total energy in the image, which should be unity, within the limitation of round off error inherent to the computer being used. Running *check_norm.c* on M1_e0n2w128s2.asc after energy normalization resulted in a total energy value of .999974.

A second test of energy normalization was also run using the correlator. A test image consisting of a solid white square on a black background was generated and energy normalized. This image was then run through the correlator and the correlation peak found. If the energy normalization worked as expected, the correlation peak should be unity. A test image whose known center is at pixel location (57,48)

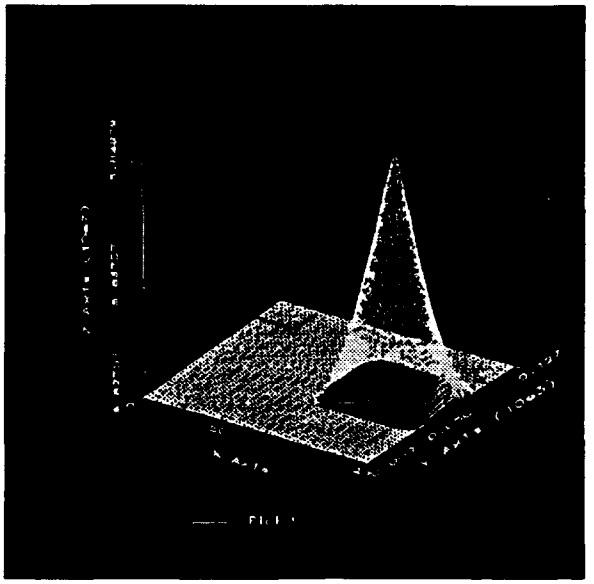


Figure 10. 3-D Test Image Correlation Plane

```
Image Statistics for File Name: /tmp/vctorKRAa17713

******* Statistics for Band #0 ******

  Mean: 5.82874e+07
  Variance: 1.01875e+09
  Std. Dev: 31917.9
  RMS:  5.82874e+07
  Peaks:
    High: 5.84879e+07 at (X,Y) location: (98,57)
    Low : 5.82757e+07 at (X,Y) location: (0,84)
  Total integral under the image: 9.54981e+11
  Positive part of integral under the image: 9.54981e+11
  Negative part of integral under the image: 0
  Contributing points: 16384
  Number of positive points in image: 16384
  Number of negative points in image: 0
  Skewness:  3.29355
  Kurtosis:  11.06
```

Figure 11. Sensitivity Test Image Statistics

was energy normalized and correlated. Referencing Figure 12, the correlation peak is 1.00003 located at location (57,48) as expected.

```
Image Statistics for File Name: /tmp/vctorKAAa15920

******* Statistics for Band #0 ******

  Mean: 0.0732447
  Variance: 0.0272194
  Std. Dev: 0.164983
  RMS:  0.180506
  Peaks:
    High: 1.00003 at (X,Y) location: (57,48)
    Low : -3.77576e-08 at (X,Y) location: (24,39)
  Total integral under the image: 1200.04
  Positive part of integral under the image: 1200.04
  Negative part of integral under the image: -1.66304e-05
  Contributing points: 16384
  Number of positive points in image: 14389
  Number of negative points in image: 1995
  Skewness:  2.66561
  Kurtosis:  6.90994
```

Figure 12. Energy Normalization Test Statistics

4.4 Image Correlation

The various versions of the images, both energy normalized and raw, were correlated using the C program *correlate.c* located in Appendix D.5. Correlate.c is simply a program that executes the C Shell program *correlate* for each image. This C Shell, located in Appendix D.6, executed the appropriate KHOROS routines to find the correlation of the image with a cutout of the target from that image, using the fast Fourier transform routine inherent to KHOROS. The local energy normalization of a cutout target with its mother image was accomplished using the C program *localnorm.c* found in Appendix D.15. Standard energy normalization was accomplished with the C program *enorm.c* and the C shell *doenorm* found in Appendices D.8 and D.9, respectively. The results from the various correlations are reported next.

4.4.1 Energy Normalized Images To *energy normalize* an image, each pixel in the image is normalized to a value between 0 and 1 by dividing by the total energy

in the image, where total energy is calculated by:

$$E_{Total} = \sqrt{\sum\sum I(x,y)^2}$$

where $I(x,y)$ is the intensity of the pixel at location (x,y) in the image. Energy normalization is designed to decrease the impact made by a spurious pixel or group of pixels with high intensity gray scale values by incorporating more of the target shape into the correlation. It decreases these spikes more than the lower values. The hope is to de-emphasize the spurious peaks to the point that the true shape of the target determines the correlation peak.

The target located in each image was extracted using the data found in Appendix E and placed in the center of a template surrounded by minimum pixel values via the program *make_template.c* found in Appendix D.12. Once the template was made, both the original image and the template were energy normalized using the C program *enorm.c* found in Appendix D.8 as called by the C Shell *doenorm* found in Appendix D.9. Once the image and template were energy normalized, they were correlated using the C Shell *new_correlate* found in Appendix D.13 as it was called by the C program *do_new_correlate.c* found in Appendix D.14. The resulting correlation statistics were examined and the location of the correlation peak and correlation minimum noted.

The results showed limited success. Of the 84 M1 tank images, three found the known target center as the correlation peak (four percent). Of the 18 airplane on runway images, all failed to find the known target center (0 percent). Finally, out of the 78 truck images, 30 were able to find the known target center (38 percent). This resulted in an overall 18 percent success rate for finding targets in raw images. For the first cut, this implies that this correlation is technique is not a viable alternative for this problem.

Analysis shows why this is true. The locations of the correlation peaks found are located in areas of "medium" pixel values. Since the templates are created with a square encompassing the target, some background pixels become part of the template. These background pixels are sufficient to skew the correlation. Heuristically, the template is made up of a mixture of light and dark pixel values, which have a median value. Correlating the template with a region of nearly homogenous pixel values which are at or near the median value of the template will result in the correlation peak being found there.

However, an interesting occurrence was noted when viewing some of the 2-D correlation planes. In some of the 2-D correlation planes, there is a region of low pixel values approximately the size of the target. Reference Figure 13 and Figure 14. Heuristically, this lends credibility to the technique of finding the location of the minimum correlation value vice the correlation peak. Using this new technique, the images are again correlated.

Using the correlation minimum, 24 of 84 tank images (29 percent), none of the airplane images (0 percent), and 28 of 78 truck images (36 percent) were located for a success rate of 52 out of 180 (29 percent). Combining the two techniques and accounting for the two tank image and eight truck image overlap, gives an overall success rate of 25 of 84 tank images (30 percent), none of the airplane images (0 percent), and 50 of 78 truck images (64 percent) located for a total success rate of 75 out of 180 (42 percent). A successful image correlation is shown in Figures 15 and 16. An unsuccessful image correlation is shown in Figures 17 and 18.

A second energy normalization technique, called *local energy normalization* was then used. Each image was read in and its target extracted using the truth data. The target pixels were energy normalized and stored in the template. The template was then overlayed at all possible positions in the original image, including the edges by using wrap around. At each position, the pixels under the template were locally energy normalized and stored. The energy normalized image pixels were

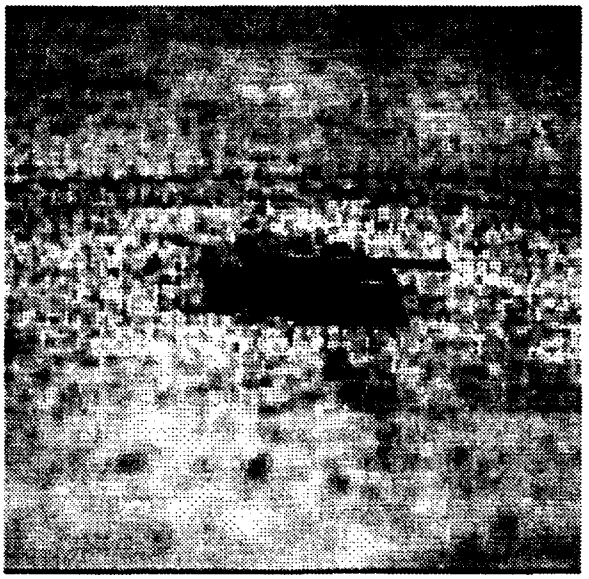


Figure 13. Original Image

Figure 14. Local minimum Correlation Plane. Notice the large group of black pixels at and around the center of the known position of the tank.

Figure 15. Original Image

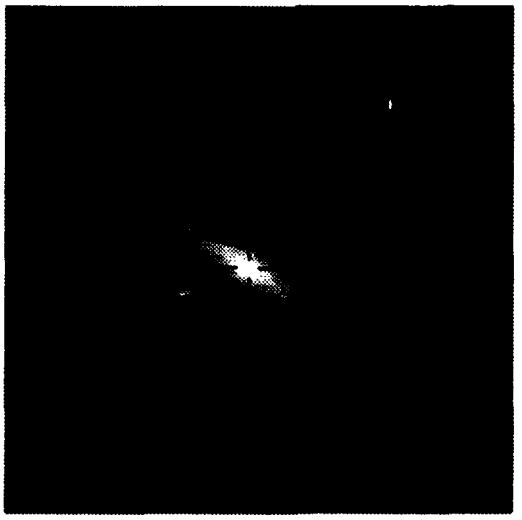


Figure 16. Raw Image 2-D Correlation Plane. The cross-hairs indicate the location of the target found by correlation.

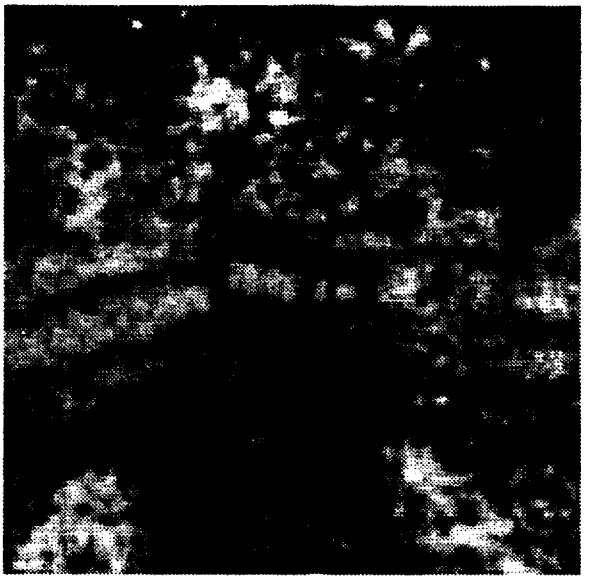


Figure 17. Original Image

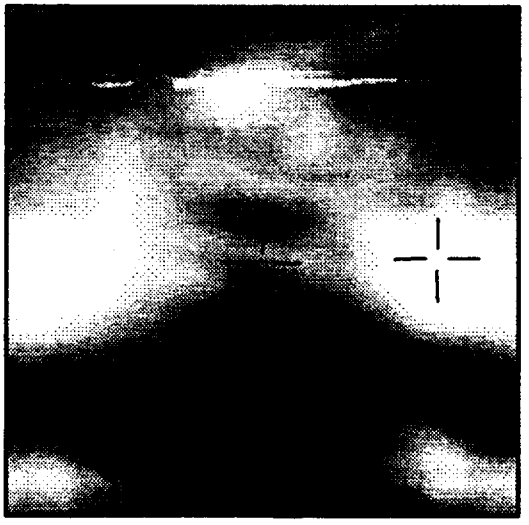


Figure 18. Raw Image 2-D Correlation Plane. The left-most cross-hairs indicate the location of the actual target and the right-most cross-hairs indicate the target location found by the correlation.

then multiplied point-by-point by the template pixels and summed. The total sum became the correlation value at that position in the image. This algorithm was implemented by the C program *localnorm.c* found in Appendix D.15.

The results from local energy normalization were encouraging. All of the 180 images found the known target center. Moreover, the correlation peak was found to be unity in each case, as was expected. Examples of an image and its 2-D correlation plane are shown in Figure 19 and Figure 20. This technique is analogous to assuming a perfect set of templates available to the correlator for target location. The results show that under this assumption, correlation is a viable technique when some means of generating good templates, possibly via a model based approach, is available.

The next logical step in this process was to assume less than perfect templates available to the correlator. All 180 images were inspected to determine a set of templates that represent the target at various scales and orientations. With this, 14 templates were chosen: four M1 tank, three F-16 airplane, one A-10 airplane, and six truck templates. The 14 chosen templates are shown in Figure 21. With the limited number of airplane images in the image set, all airplane results will be reported together.

Once the templates were chosen, like images were correlated with like templates. For example, the entire set of 84 M1 tank images were local energy normalized and correlated by the four M1 tank templates, which were also local energy normalized. The airplane and truck images were also correlated with the airplane and truck templates, respectively. This technique is analogous to a situation where the correlator is searching for a target of known type in an image, with the best resulting correlation peak denoting its location and orientation. Once the images were thus processed, each of the correlation planes for each image were inspected for the location of the correlation peak. The highest of these correlation peaks and its location were noted. The success/failure was based on correlation peak location and the template from which it resulted. If the location of the correlation peak was

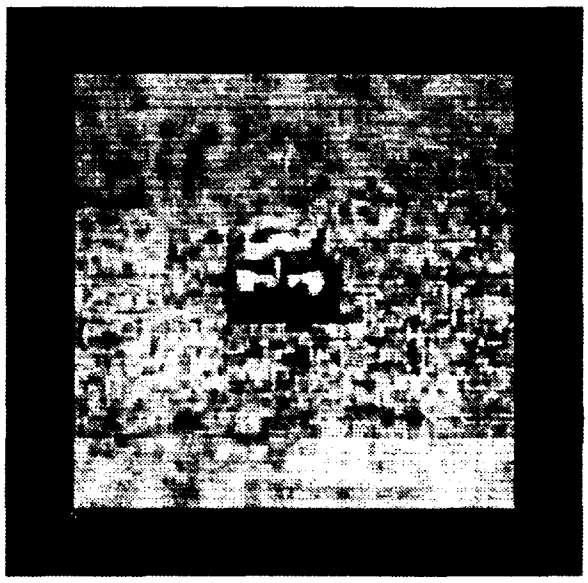


Figure 19. Original Image

Figure 20. Local Energy Normalization 2-D Correlation Plane. Notice the high-intensity pixel at the center of the tank as expected.

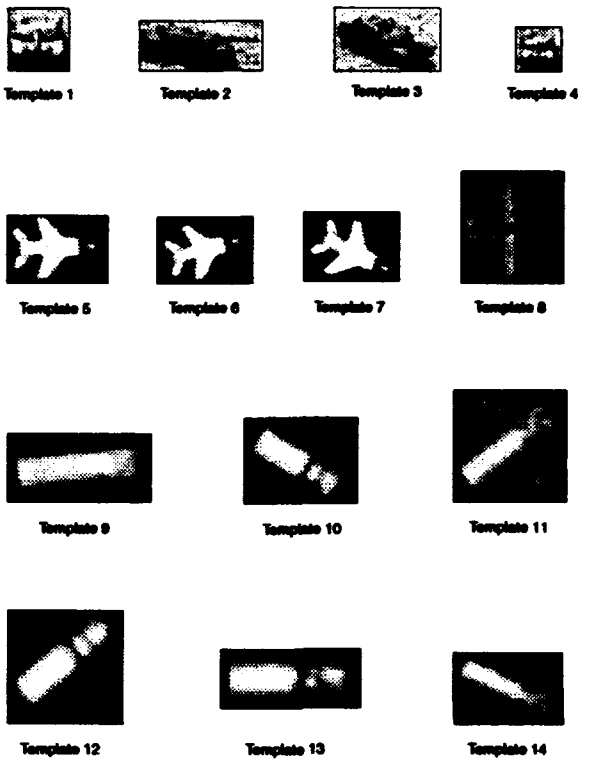


Figure 21. Correlation Template Set.

located a distance greater than half the diagonal of the box encircling the target, the correlation was deemed a failure due to the distance criteria. Similarly, if the correlation peak resulted from a target at a substantially different orientation or scale than what is in the template, the correlation was deemed a failure due to the target identification criteria. Figures 22 and 23 represent a successful correlation of a template with another image. Figures 24 and 25 represent a correlation that failed both the distance and target identification criteria.

The results from this were encouraging. Of the 84 M1 tank images, 12 failed due to the distance criteria, with an additional five failures due to the target identification criteria. This results in 67 out of 84 images correctly located, for a success rate of 80

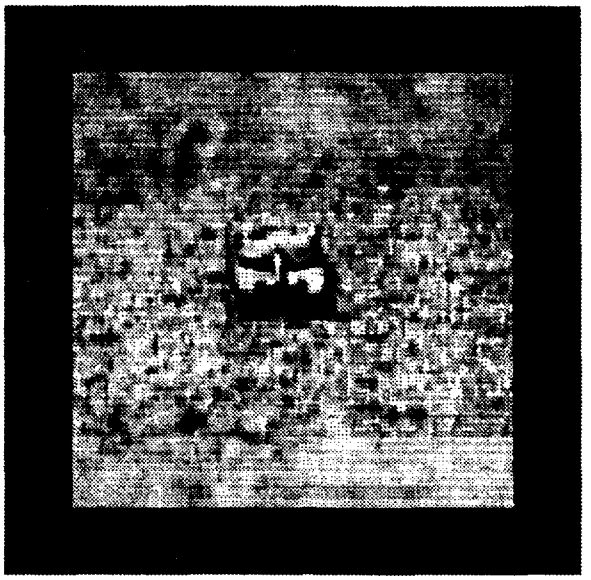


Figure 22. Original Image

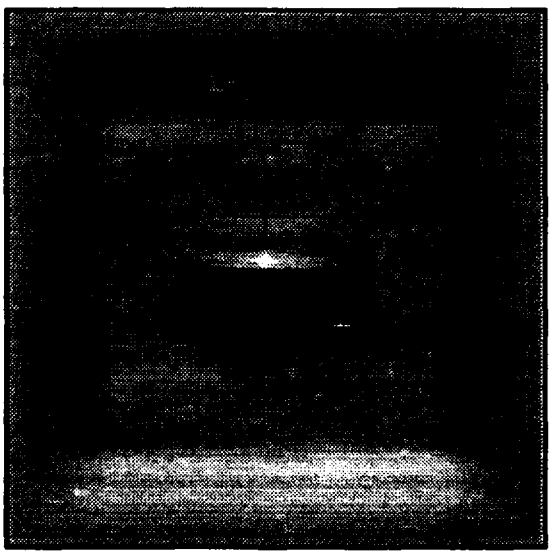


Figure 23. Local Energy Normalization 2-D Correlation Plane. Notice the high-intensity pixel at the center of the tank as expected.

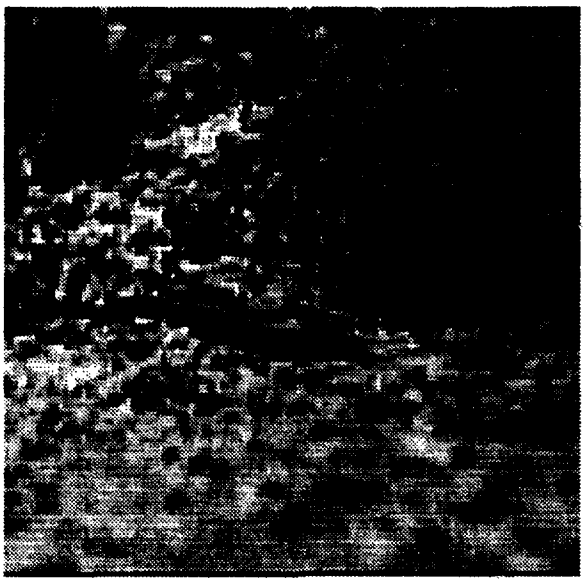


Figure 24. Original Image

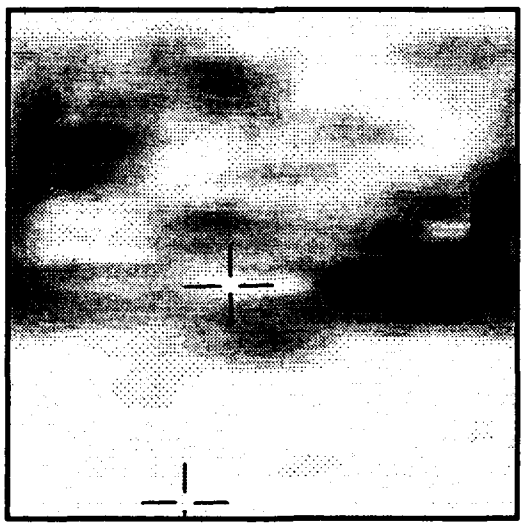


Figure 25. Local Energy Normalization 2-D Correlation Plane. The upper-most cross-hairs indicate the true location of the target, while the lower-most cross-hairs indicate the location found by correlation.

percent. Of the 18 airplane images, two failed the distance criteria with an additional two failing the target identification criteria, resulting in 14 out of 18 images correctly located and a success rate of 78 percent. Finally, of the 78 truck images, five failed the distance criteria, with an additional 11 failing the target identification criteria, resulting in 62 out of 78 successfully located and a 79 percent success rate. The overall rate is 143 out of 180 for a success rate of 79 percent.

The final step in the local normalization process was to correlate all 180 images with all 14 templates and note the highest correlation peak of the resulting 14 correlation peaks found for each image. This peak, its location, and the template from which it came were used for the success/failure determination. Using this technique, the results dropped significantly. Out of the 84 M1 tank images, 65 failed the target identification criteria with an additional two failing the distance criteria, for a success rate of 17 out of 84 (20 percent). For the airplane images, 10 failed the target identification criteria and two failed the distance criteria for a success rate of six out of 18 (33 percent). Out of the 78 truck images, 63 failed the target identification criteria with one failing the distance criteria for a success rate of 15 out of 78 (19 percent). The overall success rate of this technique is 38 out of 180 (21 percent). An example of a correlation failing both the distance criteria and target identification criteria is found in Figures 26 and 27.

Inspection of these results showed an interesting trend. A total of 141 images found the best correlation peak when matched with a template made from an A-10 airplane. However, there are only four images in the entire image set that contain an A-10. Clearly, this template was skewing the results. Perhaps removing this template would leave a better set of templates and show an increased success rate.

The previous correlation was again run with the exception that the A-10 template was removed from the template set. The results were no more encouraging. Another template, one with an M1 tank at a smaller scale, was identified as the best match 98 out of 180 times. Removing this template also resulted in another M1

Figure 26. Original Image

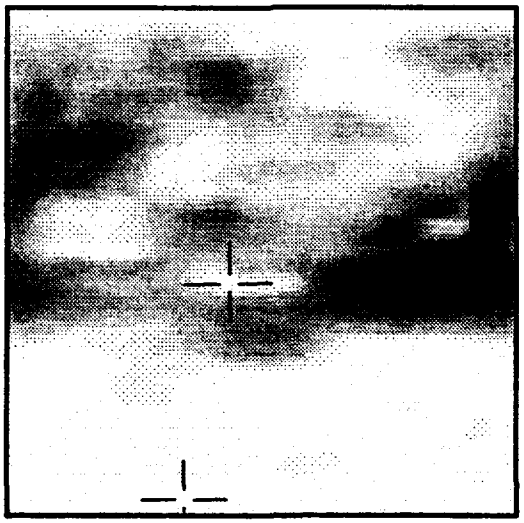


Figure 27. Local Energy Normalization 2-D Correlation Plane. The upper-most cross-hairs indicate the true location of the target, while the lower-most cross-hairs indicate the location found by correlation. This correlation also failed to identify the target correctly.

tank template dominating. Continually removing dominant templates as they appear soon lead to a case where all M1 templates had been removed, ensuring that the 84 M1 images would fail the target identification criteria. Thus, removing dominant templates did not lead to a better set of templates.

Now that the energy normalized and local energy normalized images have been investigated, correlation of the raw images was examined to determine if the success rate increases.

4.4.2 Raw Images Each raw image was correlated with a template made from the cutout target found in it. The target in the image was located and a box encircling the target noted. This box containing the target was extracted from the image and placed in the center of a 128 x 128 template, with pixel values of zero filled around it.

Of the 84 M1 tank images, three found the known target center as the correlation peak (four percent). Of the 18 airplane on runway images, all failed to find the known target center (0 percent). Finally, out of the 78 truck images, 30 were able to find the known target center (38 percent). This resulted in an overall 18 percent success rate for finding targets in raw images. Again, for the first cut, this implied that conventional correlation is not a viable alternative for this problem.

However, the same region of low pixel values was again noted as it was in the correlation of the energy normalized images. Reference Figure 28 and Figure 29. Again, the images were correlated and the technique of finding the location of the minimum correlation value was used.

Using the correlation minimum to find the location of the target, 24 of the 84 M1 tank images were able to find the known target center (29 percent). Of the 18 airplane on runway images, all again failed to find the known target center (0 percent). Of the 78 truck images, 28 found the center (36 percent). This resulted

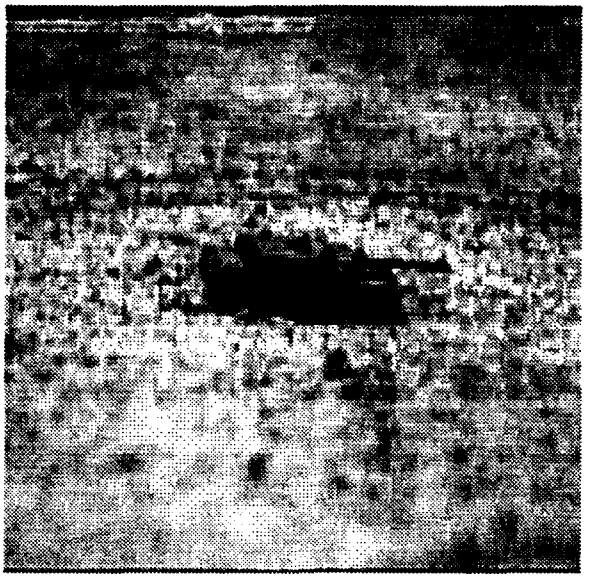


Figure 28. Original Image

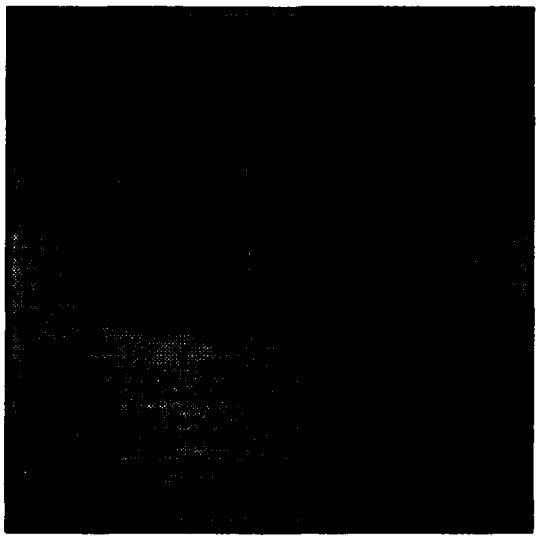


Figure 29. Local minimum Correlation Plane. Notice the large group of black pixels at and around the center of the known position of the tank.

in an overall 29 percent success rate for finding targets in the raw images when searching for the minimum correlation value.

Comparing the successes of the two techniques, there was an overlap in successful tank correlations of two images, and an overlap of successful truck correlations of eight images. That is, in 10 images, both the correlation peak and correlation minimum were able to locate the target. Combining the success rate of both techniques and taking into account this overlap, 25 of 84 M1 tank images (30 percent), 0 of 18 airplane images (0 percent), and 50 of 78 truck images (64 percent) have been successfully located, for an overall success rate of 42 percent. Comparing these results with those found with energy normalization show that the same images were successful for both the correlation peak and correlation minimum techniques. As such, other rules needed to be developed to increase the utility of correlation as a target location and identification system.

Inspection of the remaining, unsuccessful 2-D and 3-D correlation planes lead to possible heuristic rules or techniques to find and identify targets. These techniques follow.

The first heuristic technique involved examining the 3-D correlation plane for the existence of a secondary peak. A secondary peak is a large, well defined peak, whose maximum correlation value is not the correlation peak. Using the secondary peak as a rule for target location is already documented by Troxel (20) in his research. Reference Figure 30 and Figure 31. Searching the 3-D correlation plane for secondary peaks, additional targets were located. Specifically, no additional tanks (0 percent), 10 of 18 additional airplanes (56 percent), and five of 78 additional trucks (six percent) were located. The overall success rate for this technique was eight percent. Combining the result of this technique with the previous results gave success rates of 25 of 84 tank images (30 percent), 10 of 18 airplane images (56 percent), and 55 of 78 truck images (71 percent) for an overall success rate of 50 percent.

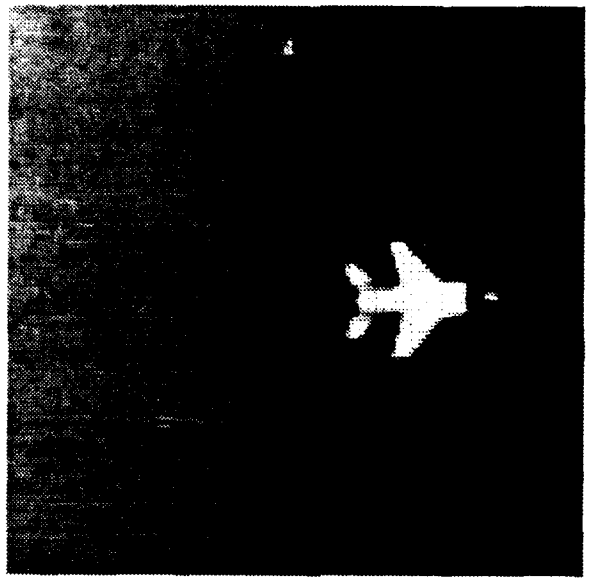


Figure 30. Original Image

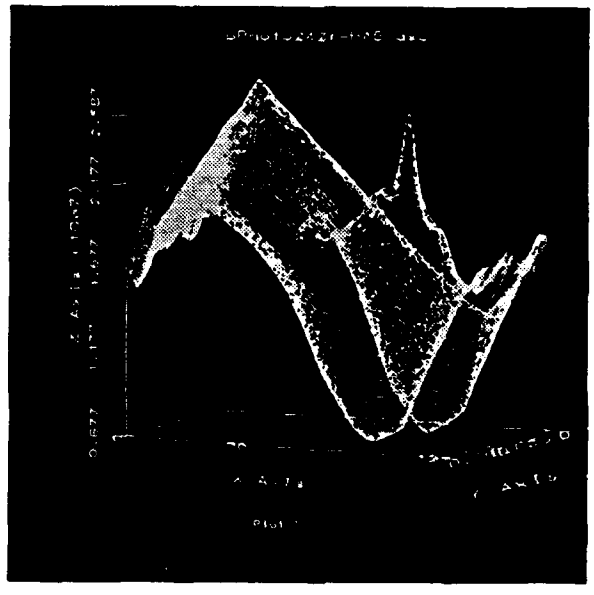


Figure 31. Secondary Peak. The peak on the right side of this graph coincides with the known center of the airplane.

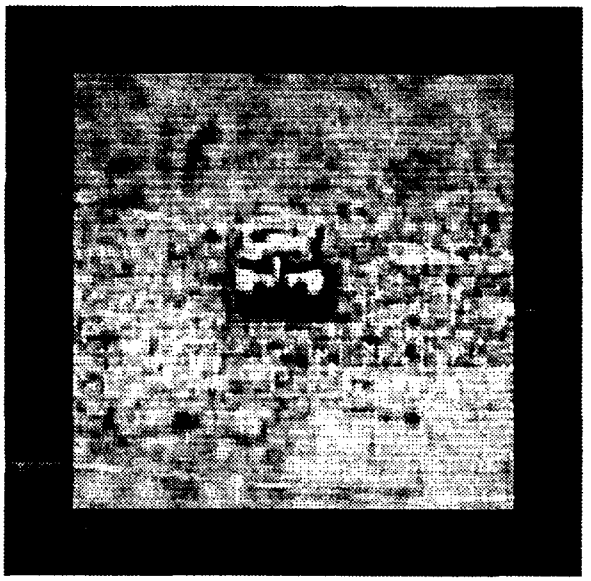


Figure 32. Original Image

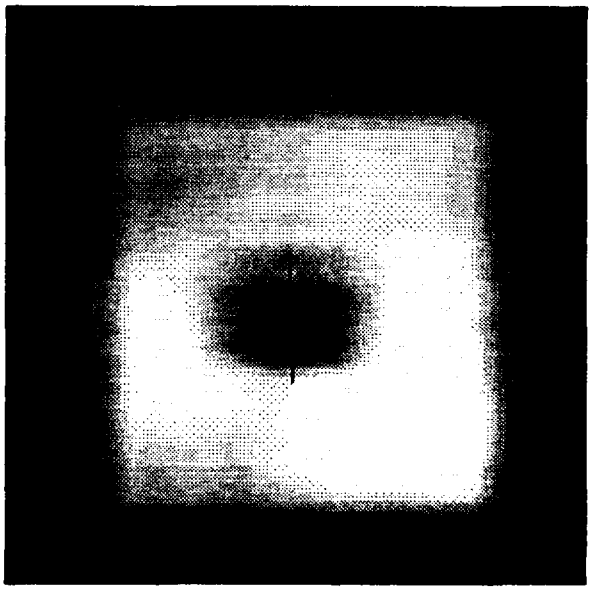


Figure 33. Heuristic Large Local Minimum Region. Notice the dark region of low pixel values at the known location of the tank.

The second heuristic technique involved searching the 2-D correlation plane for a lone region approximately the known size of the target to see if a region of small pixel values is present. Reference Figure 32 and Figure 33, there is a dark region in the area of the tank position. Using this rule, 16 of 84 tank images (19 percent), none of the airplane images (0 percent), and 14 of 78 truck images (18 percent) were located for a success rate for this technique of 17 percent. The individual success rates climbed to 41 of 84 tank images (49 percent), 10 of 18 airplane images (56 percent), and 69 of 78 truck images located (88 percent). The overall success rate has now climbed to 120 of 180 targets located (67 percent).

Another heuristic technique searched the 2-D correlation plane for a small region of low pixel values hidden among many such small regions in the image. Referencing Figure 34 and Figure 35, there is a dark region in the area of the tank position, however, it is not the only dark region present. Using this rule, 19 of 84 tank images (23 percent), no airplane images (0 percent), and no truck images (0 percent) were located. This technique's success rate is 11 percent. Overall, 60 of 84 tank images (71 percent), 10 of 18 airplane images (56 percent), and 69 of 78 truck images (88 percent) have been located. The overall success rate has now risen to 77 percent.

The final heuristic technique searches the 2-D correlation plane for a small pinpoint of high pixel values among a dark region. Reference Figure 36 and Figure 37, we can see that such a pinpoint does exist. Using this rule, 23 of 84 tank images (27 percent), three of 18 airplane images (17 percent), and 2 of 78 truck images (three percent) were located.

Combining the results from all techniques used to correlate the raw images resulted in 83 of 84 tank images located (99 percent), 13 of 18 airplane images located (72 percent), and 71 of 78 truck images located (91 percent). This lead to an overall success rate of 167 out of 180 (93 percent). Using all of the heuristic

Figure 34. Original Image

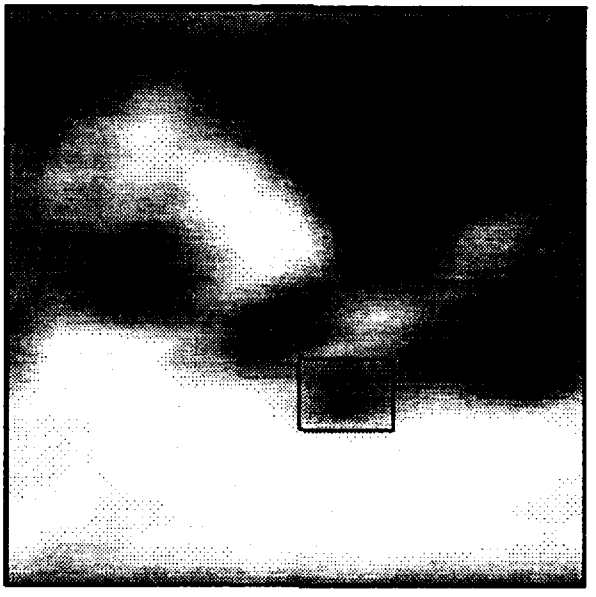


Figure 35. Heuristic Small Local Minimum Region. Notice the region enclosed in the box at the location of the tank.

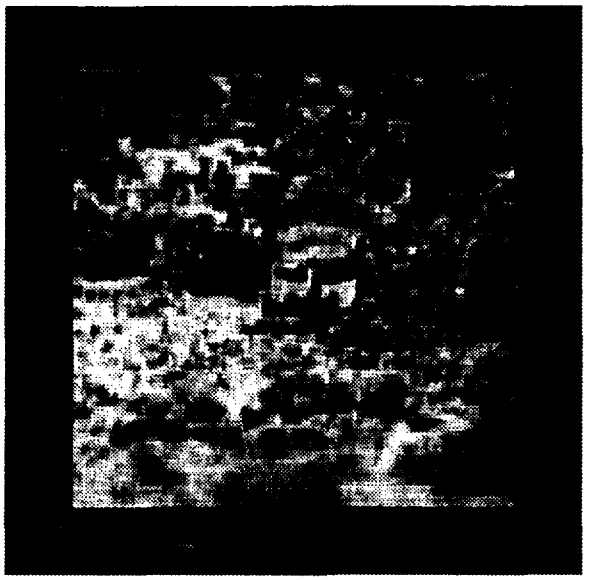


Figure 36. Original Image

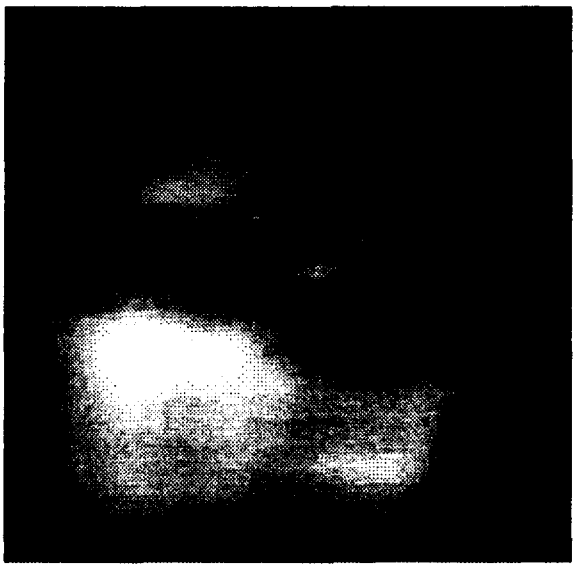


Figure 37. Heuristic Small Pinpoint Region

rules along with the correlation peak and correlation minimum, a rule-based expert system can be developed using correlation.

The overall flowchart showing all of the different correlation techniques is shown in Figure 38.

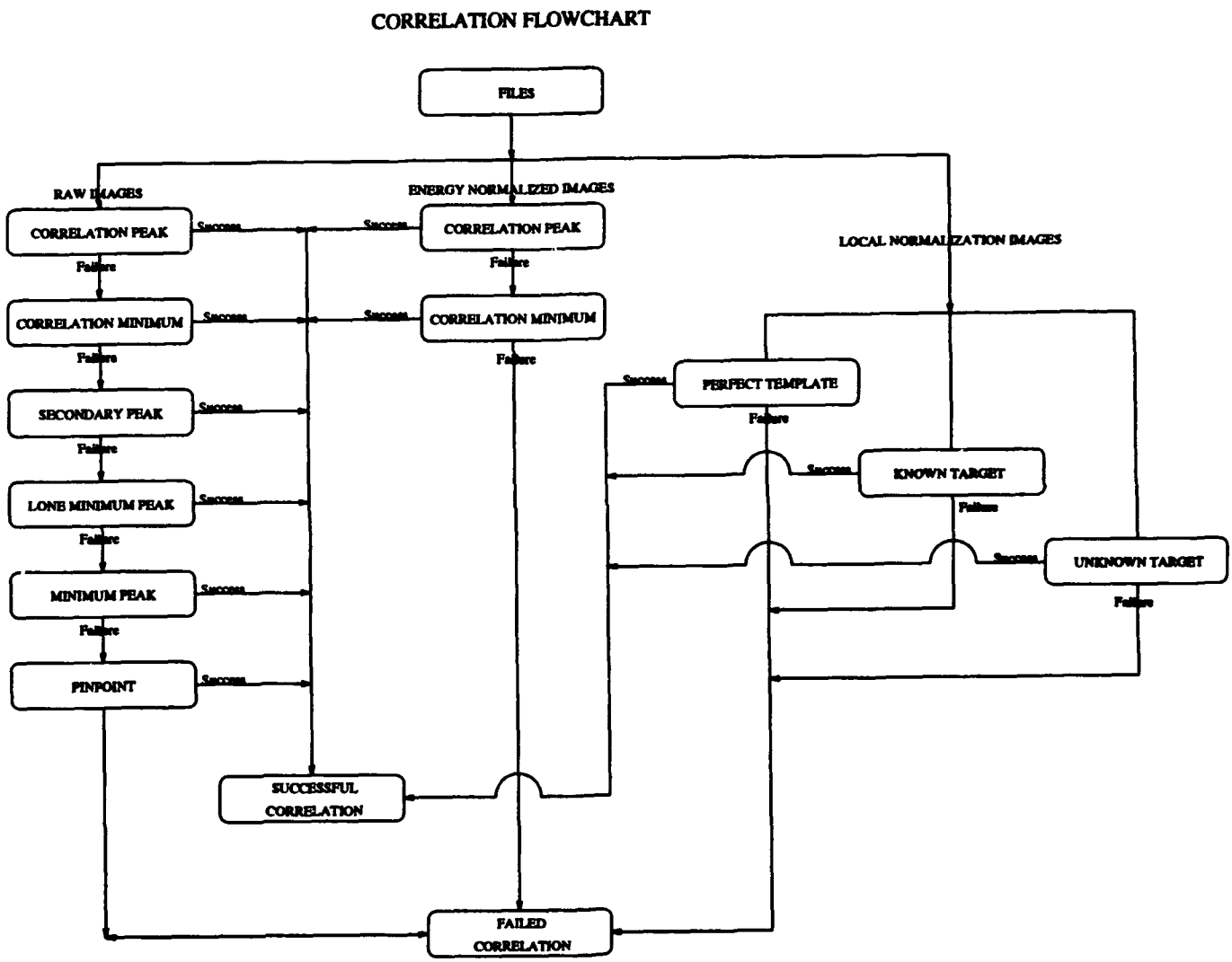


Figure 38. Correlation Flowchart

4.5 Summary

This chapter presented the results achieved by the various correlation routines and rules. Local energy normalization was used to pre-process the set of images before correlation. When a perfect template was assumed, the correlation success rate was 100conclusions reached as a result of this research effort.

V. Conclusions

This chapter briefly summarizes the results reported in Chapter IV.

5.1 Research Summary

The goal of this effort was to determine the viability of correlation as an autonomous target recognizer in visible light terrain board imagery. The images used were a representation of low light level visible imagery. Both raw, energy normalized, and locally energy normalized images were correlated using KHOROS routines and the correlation peak and correlation minimum were noted and compared to the known target location. Heuristic rules were also developed to facilitate target location.

The first part of this effort involves pre-processing the images via energy normalization. Each image, and the template made from that image, were energy normalized before correlation. Once the image was correlated with its template, the location of the correlation peak and correlation minimum were compared against the known target centers and the success rate of each was determined. When using the correlation peak, 3 0f 84 tank images (four percent), 0 of 18 of the airplanes images (0 percent), and 30 of 78 truck images (38 percent) were located, for a success rate of 33 out 180 (18 percent). When the correlation minimum was used, the success rate climbed. Of the 84 tank images, 24 targets were located (29 percent), 0 of the 18 airplane images were located (0 percent), and 28 of 78 truck images (36 percent) for a success rate of 52 out of 180 (29 percent).

Combining the energy normalization success rates for both the correlation peak and correlation minimum, and accounting for overlap between the two techniques of two tank images and eight truck images, gives success rates of 25 of 84 tank images (30 percent), 0 of the 18 airplane images (0 percent), and 50 of 78 truck images (64 percent) for an overall success rate of 75 of 180 (42 percent).

Another pre-processing technique, called local energy normalization was used to test the effectiveness of the correlator. A sequence of three experiments were run with increasing level of difficulty. The first experiment involved extracting a template from an image and then local energy normalizing both the template and the image and correlating. This is analogous to assuming that the correlator has a set of perfect images available for target location and identification as well as proving the integrity of the correlator. As expected, all 180 images found the correlation peak at the known center of the target, with the correlation peak equal to unity.

The second experiment involved identifying a set of templates that represent the image set in terms of scale and orientation of all the possible targets. Once identified, M1 tank images were correlated with M1 tank templates, airplane images were correlated with airplane templates, and truck images were correlated with truck templates. The largest correlation peak found for each image was scored against two criteria, the distance criteria and the target identification criteria. Failure of one or both criteria resulted in a failed correlation. The results were encouraging: 67 of 84 M1 tank images, 14 of 18 airplane images, and 62 of 78 truck images were successfully located for an overall success rate of 143 of 180 (79 percent).

The final experiment correlated all 180 images with all 14 templates. The largest of the 14 correlation peaks found for each image was again checked against the two criteria and the success or failure determined. Results were significantly lower. Out of the 84 M1 tank images, 17 were located for a success rate of 20 percent. A total of six of the 18 airplane images were located, for a success rate of 33 percent. Of the 78 truck images, a total of 15 were located, for a success rate of 21 percent. The overall success rate of this technique was 21 percent.

The second part of this effort involves correlating the raw images and noting the magnitude of the correlation peak and correlation minimum as well as their location. Using only the location of the correlation peak, the success rate was extremely low. Of the 84 M1 tank images, only three were located, for a success rate of four percent.

Of the 18 airplane images, all failed to locate the known target. Finally, of the 78 truck images, only 18 were located, for a success rate of 23 percent. The overall success rate of using only the correlation peak was an abysmal 12 percent.

Using only the location of the correlation minimum also led to a low success rate. Of the 84 tank images, 24 were located (29 percent). Again, none of the airplanes were successfully located. Finally, of the 78 truck images, 27 were successfully located (35 percent). The overall success rate of using only the correlation minimum led to an overall success rate of 28 percent.

Also, a set of heuristic rules were developed to enhance the effectiveness of the correlator. The success rate for the first heuristic rule, using the presence of a secondary peak, led to the following success rates: no tanks located (0 percent), 10 airplanes out of 18 located (13 percent), and five trucks out of 78 (six percent) located. The overall success rate for using the location of the secondary peak for target location was eight percent.

The second heuristic rule developed was to search the 2-D correlation plane for a lone region of low pixel values approximately the size of the known image. Using this rule, the success rates were: 16 of 84 tanks located (19 percent), none of the 18 airplane images, and 27 of 78 truck images (35 percent). The overall success rate for this technique was 24 percent.

Another heuristic rule developed was to again search the 2-D correlation plane for a region of small pixel values. Using this rule, the success rates were: 19 of 84 tank images (23 percent), no additional truck images or airplane images located. The overall success rate for this rule was 11 percent.

The final heuristic rule involved searching the 2-D correlation plane for a pinpoint of relatively high pixel values among a region of relatively low pixel values. Using this rule, the success rates were: 23 of 84 tank images (27 percent), three of

18 airplane images (17 percent), and 2 of 78 truck images (three percent) located. The overall success rate for this rule was 16 percent.

Combining the results from all of the heuristic rules with the results from the correlation peak and correlation minimum lead to individual success rates of: 83 of 84 tank images (99 percent), 13 of 18 airplane images (72 percent), and 71 of 78 truck images (91 percent) located. The final, overall percent rate for all 180 images was 93 percent for correlating raw images. These results show that correlation can be used as the basis for a rule-based expert system for autonomous target recognition.

5.2 Conclusion

The results of this research effort show that correlation is indeed a ***viable*** technique for locating targets in visible light, terrain board imagery. Classical correlation searches the correlation plane and reports the maximum value as the location of the target. However, this by itself was not shown to be a viable target locator. Other rules needed to be developed. Given these types of images, the rules for successfully locating the targets were not known ***a priori***, however they were easily found. When all of the various rules are taken as a group, correlation is indeed a viable technique for autonomous target location and recognition. These various rules form the basis of a rule-based expert system.

Appendix A. *Loading the Imagery*

Both sets of data arrived on Sun tapes. To unload these onto Scgraph, use the tape drive located in Room 132 of Building 640. First, create a subdirectory using the "md" command. Then enter this subdirectory by using the "cd" command. Once in the appropriate subdirectory, load the tape into the tape drive. Since the data on the tape is stored in the "tar" format, unload the tape by typing "tar xvf /dev/rst0", which tells the machine that tar formatted data will be read from a scuzzy tape (rst) from device 0 (the tape drive).

As files are unloaded, their names will scroll on the screen. The total time to unload a tape is around 15 or 20 minutes depending on how much is stored on it. Once the tape is finished unloading, the tape drive will automatically rewind to the beginning. Remove the tape from the tape drive and keep handy for emergencies.

The data was unloaded into various sub-subdirectories, which should be left as is. *ANVIL* expects files to be in certain subdirectories, and to change file locations is to tempt fate.

Appendix B. *HIPS Format*

The HIPS format is new to AFIT. It consists of a header followed by the actual binary gray scale data. A description of the header information, taken from the *ANVIL Software User's Manual*, follows.

The HIPS image format description:

```
blank string
blank string
number of frames(always 1)
blank string
rows (128 or 512)
cols (128 or 512)
bits per pixel (always 8)
bit packing (always 0)
pixel format (always 0 and the image must be bytes)
blank string
blank string
.(always a period)
```

Appendix C. *Image Conversions*

Given the C code to convert binary to floating point gray scale values, it was a simple matter to put in a section at the beginning to read in, and discard, information up to the period that delineates the header. Once this period is read in, the program continues on as originally written to perform the conversion. This program, titled *HIPS2float.c*, is listed in Appendix D.1. Another useful tool is a C Shell to do a batch conversion of all files with the *.hips* extension. This C Shell, titled *doHIPS2asc* and found in Appendix D.2, will read in each file with the *.hips* extension and convert to a floating point ascii value representation. It should be noted here that the various programs and routines may call this representation either ascii or floating point. These can be considered completely interchangeable for our purposes.

Once the files are in the ascii format, they can be converted to any number of file formats, including gray scale, rle, and eps formats. The routines to do this are resident on the Silicon Graphics machines and are easily accessible. These programs have been linked into one C Shell program so that the file can be left in whatever format necessary by simply commenting out the appropriate lines. This C Shell, called *rawenzel*, is listed in Appendix D.3. Included in this file is the get4d routine which will display the image onto the screen so that you can see it. The following is a description of the functions and programs called by the *rawenzel* C Shell.

- HIPS2float infile outfile.asc – this strips off the header information from the infile and converts the HIPS formatted image to ascii format.
- float_gray infile.asc outfile.gra – this converts ascii files to gray value files
- graytorle -o outfile.rle 128 128 infile.gra – takes the gray file, of dimension 128 x 128 pixels, and converts to rle format
- rleflip -v infile.rle outfile.rle – This will flip upside down rle images to right side up images. This step is unnecessary if the image is already correctly displayed

- get4d infile.rle – this will display the rle file to the screen for us to see
- crop 0 32 127 95 infile.rle > outfile.rle – crops the original from 128 x 128 to 128 by 64 by cutting off the top and bottom quarter. The cropped file will have a total of 8192 bytes. The parameters in the routine call are min-x min-y max-x and max-y respectively.
- rleprint infile.rle > outfile.asc – converts the rle file to ascii
- float_gra infile.asc outfile.gra – puts the file in final form for processing.
- rle2eps $maketemp.rle 2.5 – converts an rle file to eps for output in documents

C.1 Upside-down Image Conversion

Should any displayed image be upside down, it can be put right- side up by using the rleflip program. Notice that the input to this program is the rle version of the file and the output is also a rle file. This routine is part of the *rawenzel* C Shell and can be invoked by removing the # in front of it.

C.2 KHOROS Conversion

HIPS formatted files, once converted to their ascii representations by the *HIPS2float* program, are ready to be processed by KHOROS. If the image is up-side down, KHOROS will still process it as always, however it is tougher on the human operator. This can be rectified by the *rleflip* program.

Appendix D. *C Programs and C Shells*

D.1 HIPS2float.c

```
/*  HIPS2float.c  */

#include <stdio.h>
#include <math.h>
#define skip_line gets(junk)
#define loopi(stopi) for(i=0;i<(stopi);i++)
#define loopj(stopj) for(j=0;j<(stopj);j++)
#define loopabj(startj, stopj) for(j=(startj);j<=(stopj);j++)
#define loopabi(starti, stopi) for(i=(starti);i<=(stopi);i++)
#define loopk(stopk) for(k=0;k<(stopk);k++)
#define loopij(A,B) for(i=0;i<(A);i++) for(j=0;j<(B);j++)
#define fskip_line(A) fgets(junk, 256, A)

main(argc,argv)
int argc;
char *argv[];
{
FILE *fout, *fin; int i, j, k, count=0;
unsigned char outval;
float inval, max, min;
char junk[256];

if(argc != 3){
  printf("!!! The command line should be !!!:\n\n
          Terrain_ASC infile outfile\n\n");
```

```
  exit(0);
  }
/******************  Set Up Files  ************************/

if ((fin=fopen(argv[1],"r")) == NULL){
  printf("I can't open the input file"); exit(-1); }

if ((fout=fopen(argv[2],"w")) == NULL){
  printf("I can't open the output file"); exit(-1); }

/** skip header information  **/

while( fgetc(fin) != '.' )
  {
  fskip_line(fin);
  printf("count = %d\n", count++);
  }

fskip_line(fin);
/****  Skip last carriage return after "." ****/

/*  get, convert and write data */
while( (inval = (float) fgetc(fin)) != EOF)
   fprintf(fout,"%f\n",inval);

fclose(fin);
fclose(fout);
printf("\n I'm Done!\n");
```

```
}
/*****************   End of Main   ********************/
```

D.2 doHIPS2asc

```
#!/bin/csh foreach file(*.hips)
set temp = $file:r
HIPS2float $file.hips $file.asc
end
echo "I'm all done converting .hips files to .asc files"
```

D.3 rawenzel

```
#!/bin/csh
if ($1 == "" || $2 == "" || $3 == "" || $4 == "") then
echo "  ";
echo "The proper format is:"
echo " "
echo "rawenzel filename rootname xsize ysize ";
echo "  ";
exit -1
endif
set maketemp = $2:r
HIPS2float $1 $maketemp.asc
float_gray $maketemp.asc  $maketemp.gra
graytorle -o $maketemp.rle $3 $4 $maketemp.gra
# rleflip -v $maketemp.rle > f$maketemp.rle
```

```
get4d f$maketemp.rle
# crop 0 32 127 95 f$maketemp.rle > $maketemp.rle
# get4d $maketemp.rle
# rm $maketemp.gra
# rm $maketemp.asc
# rleprint $maketemp.rle > $maketemp.asc
# float_gray $maketemp.asc $maketemp.gra
# rm $maketemp.asc
# rle2eps $maketemp.rle 2.5
echo "I'm all done"
```

D.4 movefile

```
#!/bin/csh
foreach file(*.gra)
set temp = $file:r
mv $file.gra $file.img
# cp $file.gra $file.img
end
echo "I'm all done converting .gra files to .img files"
```

D.5 docorrelate.c

```
/*   docorrelate.c
 *   calls C Shell correlate
 *   reads in data from autocor.dat
 */

#include <stdio.h>
```

```
main() {
  int  i,x,y,w,h,xc,yc,xo,yo;
  FILE *ipf;
  char cmd[140],ecmd[140],image[30];

  ipf = fopen("autocor.dat","r");

  for (i=0; i<180; ++i){

    fscanf(ipf,"%s%d%d%d%d%d%d%d%d", image, &x, &y, &w, &h, &xc,
&yc,
    &yo, &xo);

   sprintf(cmd, "correlate %s %d %d %d %d %d %d %d %d", image,
   x,y,w,h,xc,yc,xo,yo); printf("%d%s\n",i,cmd);

   sprintf(ecmd, "ecorrelate e%s %d %d %d %d %d %d %d %d", image,
   x,y,w,h,xc,yc,xo,yo);

 system(cmd); system(ecmd);
  }
}
```

D.6 medfilter.c

```
#include <math.h>
#include <stdio.h>
#define EXTENT 9  /*  size of image i.e 128 x 128 */
```

```
#define WINDOW 5  /* size of window, i.e 5 x 5 */

void main(argc,argv)
int argc;
char *argv[];

{
     FILE *ofp,*ifp;
     float val[EXTENT][EXTENT],outval[EXTENT][EXTENT];
     float sortval[WINDOW*WINDOW],temp,median;
     int i,j,k,l,m,n,q,r,offset,newdim,count;

     if ((ifp=fopen(argv[1],"r")) == NULL){

/*      if ((ifp=fopen("","r")) == NULL){*/
        printf("I can't open the input file");
        exit(-1);
     }

     if ((ofp=fopen(argv[2],"w")) == NULL){
/*       if ((ofp=fopen("","w")) == NULL){*/
       printf("I can't open the output file");
        exit(-1);
     }

     for (i=0; i<EXTENT; ++i){
         for (j=0; j<EXTENT; ++j){
             val[i][j]=0.0;
```

```
            outval[i][j] = 0.0;
        }
    }

    for (i=0; i<EXTENT; ++i){
        for(j=0; j<EXTENT; ++j){
           (fscanf(ifp,"%f",&val[i][j]));
        }
    }

    offset = (WINDOW-1)/2;
    newdim = EXTENT - WINDOW + 1;
/*    printf("offset is %d and newdim is %d\n",offset,newdim);*/
    count = 0;

   for(i=offset; i<(offset+newdim); ++i){
        for(j=offset; j<(offset+newdim); ++j){

           for(l=(i-offset); l<(i+offset+1); ++l){
               for(m=(j-offset); m<(j+offset+1); ++m){
           /*  printf("i is %d ,j is %d,l is %d, m is
%d\n",i,j,l,m); */

/* read values into sortval for sorting*/

                    sortval[count] = val[l][m];

 /*             printf("sortval[%d] is %f\n",count,val[l][m]); */
```

```
                count = count + 1;
                } /*  close m  */
            } /*  close l  */
             count = 0;
/*  begin bubblesort on sortval[WINDOW*WINDOW]  */

                for(q=0; q<((WINDOW*WINDOW)-1); ++q){
                   for(r=((WINDOW*WINDOW)-1); r>q; --r){
                      if (sortval[r-1] < sortval[r]){
                         temp = sortval[r-1];
                         sortval[r-1] = sortval[r];
                         sortval[r] = temp;
                      }/*  close if */
                   }/*  close r */
                }/* close q  */

/*  end bubblesort  */
/*                  for (k=0; k<(WINDOW*WINDOW); ++k){
                      printf("%f\n",sortval[k]);
                }
               printf("\n");*/

             median = sortval[((WINDOW*WINDOW)-1)/2];
/*             printf("med is %f\n\n%",median);*/
             outval[i][j] = median;

        } /*  close j */
    }/*    close i  */
```

```
    for (i=0; i<EXTENT; ++i){
        for (j=0; j<EXTENT; ++j){
            fprintf(ofp,"%f\n",outval[i][j]);
        }
    }

} /*  end of main */
```

D.7 domedfilter

```
#!/bin/csh
foreach file(*.asc)
set temp = $file:r
set outfile = $temp.med

echo $outfile
medfilter $file $outfile

end
echo "I'm all done median filtering .asc files"
```

D.8 enorm

```
/*  This will energy normalize images */

#include <stdio.h>
#include <math.h>

main(int argc,char **argv)
```

```
{
    int  i,j;
    FILE *ofp,*ifp;
    float val[128][128];
    double eval[128][128],sum;

    if(argc != 3) {
      printf("\n%s%s%s\n\n",
         "usage:  ", argv[0], " infile, outfile");
      exit(1);
    }

    for (i=0; i<128; ++i){
        for (j=0; j<128; ++j){
            val[i][j]=0.0;
            eval[i][j]=0.0;
        }
    }
    sum=0.0;

    ifp = fopen(argv[1],"r");
    ofp = fopen(argv[2],"w");

    for (i=0; i<128; ++i){
        for(j=0; j<128; ++j){
           (fscanf(ifp,"%f",&val[i][j]));
        }
    }
```

```
    for (i=0; i<128; ++i){
        for (j=0; j<128; ++j){
            sum = sum + (double)(val[i][j]*val[i][j]);
        }
    }
    sum = sqrt(sum);

    for (i=0; i<128; ++i){
        for (j=0; j<128; ++j){
            eval[i][j] = val[i][j]/sum;
        }
    }

    for (i=0; i<128; ++i){
        for (j=0; j<128; ++j){
            fprintf(ofp,"%lf\n",eval[i][j]);
        }
    }

}
```

D.9 doenorm

```
#!/bin/csh
# cd ../M1IMAGES
foreach file(*.asc)
set temp = $file
enorm $file e$file
```

```
end
echo "I'm all done enorming .asc files"
```

D.10 correlate

```
#!/bin/csh

set infile = $1
set x=$2
set y=$3
set w=$4
set h=$5
set xc=$6
set yc=$7
set xo=$8
set yo=$9

set root=$1:r
set outfile=$root.dt5


asc2viff -i ./KTRUTH/$infile -o /tmp/raw10 -fi 0 -t 0 -r 128
-c 128 -m 1 -n 1 -b 1 -d 0 -h 0 -s 0

vextract -i /tmp/raw10 -o /tmp/raw20 -x $x -y $y -w $w -h $h

vfft -i1 /tmp/raw10 -o1 /tmp/raw30 -d 0

vpad -i /tmp/raw20 -o /tmp/raw40 -r 128 -c 128 -d $yo -e $xo
```

```
-j 0 -k 0

vfft -i1 /tmp/raw40 -o1 /tmp/raw50 -d 0

vconj -i /tmp/raw50 -o /tmp/raw60

vmul -i1 /tmp/raw30 -i2 /tmp/raw60 -o /tmp/raw70

vfft -i1 /tmp/raw70 -o1 /tmp/raw80 -d 1

vtranslat -i /tmp/raw80 -o /tmp/raw90 -x 64 -y 64 -w 1

vctor -i /tmp/raw90 -o /tmp/raw100 -t 3

vstats -i /tmp/raw100 -f ./KTRUTH/$outfile -all 1 -mu 0 -var 0 -sd 0
-rms 0 -vmax 0 -xmax 0 -ymax 0 -vmin 0 -xmin 0 -ymin 0 -in 0 -pin 0
-nin 0 -pts 0 -ppts 0 -npts 0 -sk 0 -kur 0 -ent 0 -con 0 -b 1

#xprism3  -title "$infile" -imlp im132 -alpha 90.0 -theta 0.0 -i 11
/tmp/raw100
```

D.11 check_norm.c

```
/*  This will check energy normalize images */

#include <stdio.h>
#include <math.h>
```

```
main(argc,argv)
int argc;
char *argv[];

{


     int  i,j;
     FILE *ofp,*ifp;
     float val[128][128];
     double eval[128][128],sum;

     if(argc != 2) {
       printf("\n%s%s%s\n\n",
          "usage:  ", argv[0], " infile");
       exit(1);
     }

     sum=0.0;

     if ((ifp = fopen(argv[1],"r")) == NULL){
        printf(" I can't open ifp %s",argv[1]);
        exit(-1);
      }

     for (i=0; i<128; ++i){
         for(j=0; j<128; ++j){
```

```
            (fscanf(ifp,"%lf",&eval[i][j]));
            sum = sum + (eval[i][j]*eval[i][j]);

        }
    }

    printf("%f\n",sum);
}
```

D.12 make_template.c

```
/*   make_template.c  reads in truth data from autocor.dat
 *   extracts the target and places in template names
 *   "T" followed by original images name.
 */

#include <stdio.h>

main()
{
  int  i,j,k,x,y,w,h,xc,yc,xo,yo,l,m;
  FILE *ipf1,*ipf2,*opf;
  char image[60],s1[60];
  float orig_file[128][128],val[128][128];

  ipf1 = fopen("autocor.dat","r");

  for (k=0; k<180; ++k){
```

```
    fscanf(ipf1,"%s%d%d%d%d%d%d%d%d", image, &x, &y, &w, &h,
    &xc, &yc, &xo, &yo);

/*  open original image for reading */

    if ((ipf2 = fopen(image,"r")) == NULL){
        printf(" I can't open ipf2 %s\n",image);
        exit(-1);
        }

/* read in data from original image  */

    for (i=0; i<128; ++i){
       for (j=0; j<128; ++j){
          fscanf(ipf2,"%f",&orig_file[i][j]);
       }
    }

/*  backfill template array with all 0.0 values */

    for (i=0; i<128; ++i){
       for (j=0; j<128; ++j){
          val[i][j] = 0.0;
       }
    }

/* fill in template array with target pixels,
   and center (one step)  */
```

```
    l = yo;
    for (i=y; i<y+h; ++i){
       m = xo;
       for (j=x; j<x+w; ++j){
          val[l][m] = orig_file[i][j];
          m = m + 1;
       }

       l = l + 1;
    }

/*  prepare outfile name T + original image name
    and store template array */

    s1[0] = 'T';
    s1[1] = '\0';
    strcat(s1,image);

    printf("%d %s\n",k,s1);

    if ((opf = fopen(s1,"w")) == NULL){
        printf (" I can't open opf \n");
        exit(-1);
        }

    for (i=0; i<128; ++i){
       for (j=0; j<128; ++j){
```

```
            fprintf(opf,"%f\n",val[i][j]);
        }
    }

    fclose(ipf2);
    fclose(opf);
  }
}
```

D.13 new_correlate

```
#!/bin/csh

set image = $1
set template = $2
set root = $1:r
set outfile = e$root.tdat
set eimage = e$image
set etemplate = e$template

#echo $image $eimage
#echo $template $etemplate
#echo $root
#echo $outfile
#echo $eimage
#echo $etemplate
echo""

enorm $image $eimage
```

```
enorm $template $etemplate


asc2viff -i ../KTRUTH/$eimage -o /tmp/raw10 -fi 0 -t 0
-r 128 -c 128 -m 1 -n 1 -b 1 -d 0 -h 0 -s 0


vfft -i1 /tmp/raw10 -o1 /tmp/raw20 -d 0


asc2viff -i ../KTRUTH/$etemplate -o /tmp/raw30 -fi 0 -t 0
-r 128 -c 128 -m 1 -n 1 -b 1 -d 0 -h 0 -s 0

vfft -i1 /tmp/raw30 -o1 /tmp/raw40 -d 0

vconj -i /tmp/raw40 -o /tmp/raw50


vmul -i1 /tmp/raw20 -i2 /tmp/raw50 -o /tmp/raw60

vfft -i1 /tmp/raw60 -o1 /tmp/raw70 -d 1

vtranslat -i /tmp/raw70 -o /tmp/raw80 -x 64 -y 64 -w 1

vctor -i /tmp/raw80 -o /tmp/raw90 -t 1

vstats -i /tmp/raw90 -f ../KTRUTH/$outfile -all 1 -mu 0 -var 0 -sd 0
-rms 0 -vmax 0 -xmax 0 -ymax 0 -vmin 0 -xmin 0 -ymin 0 -in 0 -pin 0
```

```
-nin 0 -pts 0 -ppts 0 -npts 0 -sk 0 -kur 0 -ent 0 -con 0 -b 1

echo $eimage
echo $etemplate
rm $eimage
rm $etemplate
```

D.14 do_new_correlate.c

```
/*   do_new_correlate.c
 *   calls C shell new_correlate
 *   reads in data from autocor.dat
 */

#include <stdio.h>

main()
{
  int  i,x,y,w,h,xc,yc,xo,yo;
  FILE *ipf;
  char cmd[140],ecmd[140],image[30];

  ipf = fopen("autocor.dat","r");

  for (i=0; i<180; ++i){

    fscanf(ipf,"%s%d%d%d%d%d%d%d%d", image, &x, &y, &w, &h,
    &xc, &yc, &yo, &xo);
```

```
    sprintf(cmd, "new_correlate %s T%s", image, image);
    printf("%d %s\n",i, cmd);
    system(cmd);
   }
}
```

D.15 localnorm.c

```
/*   reads in truth data from autocor.dat
 *   opens filename stored in image for reading
 *   reads in pixel values
 *   extracts target from image using truth data,
 *   stores as template
 *   energy normalizes the image and template.
 *   performs point by point correlation
 *
 */

#include <stdio.h>
#include <math.h>


main()
{
  int  i,j,k,x,y,w,h,xc,yc,xo,yo,l,m,a,b,count;
  FILE *ipf1,*ipf2,*opf;
  char image[60],s1[100];
  float orig_file[128][128],val[128][128],window[128][128];
```

```
  double ewindow[128][128],sum,etemp[128][128],cor[128][128];

/* open the file containing the truthing data */

  ipf1 = fopen("autocor.dat","r");

  for (k=0; k<180; ++k){ /*  begin k counter */

    /*  read in truth data for each file  */

    fscanf(ipf1,"%s%d%d%d%d%d%d%d%d", image, &x, &y, &w, &h, &xc,
    &yc, &xo, &yo);

/*  use truth data to read in each image  */

    if ((ipf2 = fopen(image,"r")) == NULL){
        printf(" I can't open ipf2 %s",image);
        exit(-1);
        }
   printf("reading in %d %s\n",k,image);
    for (i=0; i<128; ++i){
       for (j=0; j<128; ++j){
          fscanf(ipf2,"%f",&orig_file[i][j]);

       }
    }

/*  extract target from image using truth data and
```

```
    find the sum of the squared pixel values */

    l = 0;
    sum = 0.0;
    count = 0;
    for (i=y; i<y+h; ++i){
       m = 0;
       for (j=x; j<x+w; ++j){
          count = count + 1;
          val[l][m] = orig_file[i][j];
          sum = sum +(double)(val[l][m]*val[l][m]);
          m = m + 1;
       }

       l = l + 1;
    }

 sum = sqrt(sum);

/* energy normalize the template  */

 for (i=0; i<h; ++i){
    for (j=0; j<w; ++j){

    etemp[i][j] = val[i][j]/sum;
    }
 }
```

```
/*  perform local normalization for all positions
    in the image of same size as the template  */

   for (i=0; i<128; ++i){   /*   over all possible  */
      for (j=0; j<128; ++j){ /*     positions in the image   */
         l = 0;
         sum = 0.0;



/* index on size  */
/* of template    */

         for (a=i-(h/2); a<i+((h+1)/2); ++a){
            m = 0;
            for (b=j-(w/2); b<j+((w+1)/2); ++b){
                window[l][m] = orig_file[BOUND(a)][BOUND(b)];
                sum = sum + (double)(window[l][m]*window[l][m]);
                m = m + 1;
            }
            l = l + 1;
         }


        sum = sqrt(sum);

/*  energy normalize window */
/*    and store in ewindow   */

        for (a=0; a<h; ++a){
```

```
            for (b=0; b<w; ++b){
                ewindow[a][b] = window[a][b]/sum;
            }
        }

        sum = 0.0;

/*  point by point multiply  */
/*  ewindow and etemp and sum */
        for (a=0; a<h; ++a){
            for (b=0; b<w; ++b){
                sum = sum + (ewindow[a][b] * etemp[a][b]);
            }
        }

/* store correlation value in cor array  */
       cor[i][j] = sum;

      }
   }

/*  prepare s1 to have name of output image
and write cor array to it  */
   s1[0] = '\0';
   strcat(s1,"cor");
   strcat(s1,image);

   printf("%s\n\n",s1);
```

```
    if ((opf = fopen(s1,"w")) == NULL){
        printf (" I can't open opf");
        exit(-1);
    }

    for (i=0; i<128; ++i){
       for (j=0; j<128; ++j){
          fprintf(opf,"%f\n",cor[i][j]);
       }
    }
    fclose(s1);
    fclose(ipf2);

  }  /* end of k counter */
} /*  end of main */
```

Appendix E. *Image Truth Data*

The following truthing data was stored in the file *autocor.dat*. The format of this file is as follows:

```
image x y w h xc yc xo yo

  image is the name of the ascii file that holds the image data

      x is the upper left x coordinate used with the KHOROS glyph vextract
      y is the upper left y coordinate used with the KHOROS glyph vextract
      w is the width in pixels used with the KHOROS glyph vextract
      h is the height in pixels used with the KHOROS glyph vextract

      xc is the x coordinate of the center pixel of the target (calculated)
      yc is the y coordinate of the center pixel of the target (calculated)

      xo is the offset of column used with KHOROS glyph vpad
      yo is the offset of row used with the KHOROS glyph vpad

M1_e0n2w128s2.asc 50 47 23 24 61 59 52 52
M1_e0n3w128s2.asc 50 48 24 24 62 60 52 52
M1_e0n5bw128s2.asc 53 49 24 24 65 61 52 52
M1_e0n5ew128s2.asc 48 59 23 23 59 70 52 52
M1_e0n5fw128s2.asc 62 64 22 21 73 74 53 53
M1_e0n5w128s2.asc 48 46 26 24 61 58 51 52
M1_e0n6w128s2.asc 49 47 25 24 61 59 51 52
M1_e0n7w128s2.asc 49 48 25 23 61 59 51 52
```

M1_e0n8w128s2.asc 48 48 26 23 61 59 51 52

M1_e0n9w128s2.asc 46 47 28 24 60 59 50 52

M1_e0p0cw128s2.asc 57 48 25 23 69 59 51 52

M1_e0p0dw128s2.asc 47 52 25 24 59 64 51 52

M1_e0p0ew128s2.asc 47 57 25 23 59 68 51 52

M1_e0p0fw128s2.asc 63 65 19 20 72 75 54 54

M1_e0p100s2w128.asc 44 50 56 23 72 61 36 52

M1_e0p105s2w128.asc 43 50 56 23 71 61 36 52

M1_e0p10bs2w128.asc 38 55 25 23 50 66 51 52

M1_e0p10cs2w128.asc 41 55 28 25 55 67 50 51

M1_e0p10fs2w128.asc 43 58 24 22 55 69 52 53

M1_e0p110s2w128.asc 44 49 56 22 72 60 36 53

M1_e0p115s2w128.asc 46 50 55 22 73 61 36 53

M1_e0p15s2w128.asc 40 46 32 28 56 60 48 50

M1_e0p1w128s2.asc 50 46 23 25 61 58 52 51

M1_e0p20s2w128.asc 40 48 34 27 57 61 47 50

M1_e0p25s2w128.asc 40 48 37 27 58 61 45 50

M1_e0p2w128s2.asc 50 48 23 23 61 59 52 52

M1_e0p30s2w128.asc 40 48 39 27 59 61 44 50

M1_e0p35s2w128.asc 41 48 42 27 62 61 43 50

M1_e0p3w128s2.asc 50 48 24 24 62 60 52 52

M1_e0p40bs2w128.asc 31 57 41 23 51 68 43 52

M1_e0p40fs2w128.asc 40 60 35 20 57 70 46 54

M1_e0p40s2w128.asc 41 48 43 26 62 52 42 51

M1_e0p45ds2w128.asc 35 55 43 25 56 67 42 51

M1_e0p45es2w128.asc 39 63 39 25 58 75 44 51

M1_e0p45fs2w128.asc 40 59 35 22 57 70 46 53

M1_e0p45s2w128.asc 42 48 45 27 64 61 41 50

M1_e0p4w128s2.asc 49 47 25 25 61 59 51 51

M1_e0p50bs2w128.asc 31 58 47 25 54 70 40 51

M1_e0p50cs2w128.asc 33 56 47 25 56 68 40 51

M1_e0p50fs2w128.asc 40 60 38 21 59 70 45 53

M1_e0p50s2w128.asc 42 49 49 27 66 62 39 50

M1_e0p55s2w128.asc 43 50 49 25 67 62 39 51

M1_e0p5bw128s2.asc 26 47 26 24 39 59 51 52

M1_e0p5fw128s2.asc 62 64 21 21 72 74 53 53

M1_e0p5w128s2.asc 50 47 25 25 62 59 51 51

M1_e0p60s2w128.asc 44 48 52 27 70 61 38 50

M1_e0p65s2w128.asc 45 48 53 27 71 61 37 50

M1_e0p6w128s2.asc 49 47 27 25 62 59 50 51

M1_e0p70s2w128.asc 45 48 55 25 72 60 36 51

M1_e0p75s2w128.asc 47 49 56 25 75 61 36 51

M1_e0p7w128s2.asc 48 47 28 25 62 59 50 51

M1_e0p80es2w128.asc 45 62 48 22 69 73 40 53

M1_e0p80fs2w128.asc 41 60 45 20 63 70 41 54

M1_e0p85ds2w128.asc 28 59 58 22 57 70 35 53

M1_e0p85es2w128.asc 44 60 47 23 67 71 40 52

M1_e0p85fs2w128.asc 42 60 45 20 64 70 41 54

M1_e0p85s2w128.asc 49 49 57 24 77 61 35 52

M1_e0p8w128s2.asc 49 45 26 26 62 58 51 51

M1_e0p90bs2w128.asc 32 52 59 22 61 63 34 53

M1_e0p90fs2w128.asc 42 60 47 19 65 69 40 54

M1_e0p90s2w128.asc 50 49 58 25 79 61 35 51

M1_e0p95s2w128.asc 41 48 58 26 70 61 35 51

M1_e0p9w128s2.asc 49 45 28 27 63 58 50 50

M1_e1n5bw128s2.asc 78 80 28 28 92 94 50 50

M1_e1n5ew128s2.asc 55 63 28 26 69 76 50 51

M1_e1n5fw128s2.asc 53 55 22 20 64 65 53 54

M1_e1p0bw128s2.asc 80 81 26 27 93 94 51 50

M1_e1p0ew128s2.asc 51 63 26 27 64 76 51 50

M1_e1p0fw128s2.asc 49 55 20 20 59 65 54 54

M1_e1p5bw128s2.asc 80 81 27 27 93 94 50 50

M1_e1p5ew128s2.asc 43 62 27 28 56 76 50 50

M1_e1p5fw128s2,asc 45 55 20 20 55 65 54 54

M1_e2n5dw128s2.asc 35 26 31 37 50 44 48 45

M1_e2n5ew128s2.asc 64 59 29 33 78 75 49 47

M1_e2n5fw128s2.asc 86 62 23 25 97 74 52 51

M1_e2p0fw128s2x16_15.asc 83 64 22 27 94 77 53 50

M1_e2p0fw128s2x17_15.asc 84 65 23 28 95 79 52 50

M1_e2p0fw128s2x19_15.asc 86 65 25 30 98 80 51 49

M1_e2p0fw128s2x6_5.asc 86 65 25 30 98 80 51 49

M1_e2p5fw128s2.asc 72 66 22 24 83 78 53 52

M1_e2p5fw128s2x16_15.asc 73 67 22 24 84 79 53 52

M1_e2p5fw128s2x17_15.asc 74 67 23 26 85 80 52 51

M1_e2p5fw128s2x19_15.asc 75 68 25 28 87 82 51 50

M1_e2p5fw128s2x6_5.asc 75 68 25 28 87 82 51 50

pPhoto242r_HA5.asc 73 50 44 32 95 66 42 48

pPhoto243r_HA5.asc 68 48 44 32 90 64 42 48

pPhoto244r_HA5.asc 72 48 44 32 94 64 42 48

pPhoto245r_HA5.asc 70 52 40 30 90 67 44 49

pPhoto246r_HA5.asc 71 39 41 30 91 54 43 49

pPhoto247r_HA5.asc 73 42 41 30 93 57 43 49

pPhoto248r_HA5.asc 68 44 43 32 89 60 42 48

pPhoto249r_HA5.asc 70 44 41 32 90 60 43 48

pPhoto250r_HA5.asc 68 42 41 32 88 58 43 48

pPhoto251r_HA5.asc 44 47 42 32 65 63 43 48

pPhoto251r_HA6.asc 49 51 36 29 67 65 46 49

pPhoto251r_HA7.asc 45 42 42 44 66 64 43 42

pPhoto252r_HA5.asc 45 48 40 30 65 63 44 49

pPhoto252r_HA6.asc 49 49 35 30 66 64 46 49

pPhoto252r_HA7.asc 45 40 42 45 66 62 43 41

pPhoto272r_HA7.asc 42 42 43 46 63 65 42 41

pPhoto281r_HA5.asc 44 48 41 30 64 63 43 49

pPhoto281r_HA7.asc 62 40 43 45 83 62 42 41

tPhoto068_HV4_1.asc 40 53 46 22 63 64 41 53

tPhoto069_HV4_1.asc 40 53 45 22 62 64 41 53

tPhoto131_HV3_1.asc 45 87 35 18 62 96 46 55

tPhoto133_HV3_1.asc 43 52 34 26 60 65 47 51

tPhoto133_HV4_1.asc 40 54 44 20 62 64 42 54

tPhoto134_HV3_1.asc 47 51 32 26 63 64 48 51

tPhoto134_HV4_1.asc 45 55 43 20 66 65 42 54

tPhoto135_CV4_1.asc 12 68 43 16 33 76 42 56

tPhoto135_HV3_1.asc 52 52 32 26 68 65 48 51

tPhoto136_CV4_1.asc 44 44 42 42 65 65 43 43

tPhoto136_HV3_1.asc 47 56 34 15 64 63 47 56

tPhoto137_HV3_1.asc 50 49 30 27 65 62 49 50

tPhoto137_HV4_1.asc 40 50 43 23 61 61 42 52

tPhoto138_HV4_1.asc 10 52 43 18 31 61 42 55

tPhoto138_OV3_1.asc 50 48 33 27 66 61 47 50

tPhoto139_LV3_1.asc 50 46 28 31 64 61 50 48

tPhoto140_HV4_1.asc 43 49 42 31 64 64 43 48

tPhoto140_LV3_1.asc 50 47 27 31 63 62 50 48

tPhoto141_CV4_2.asc 43 55 44 22 65 66 42 53

tPhoto141_HV3_2.asc 48 85 32 15 64 92 48 56

tPhoto141_HV4_1.asc 42 51 42 29 63 65 43 49

tPhoto141_LV3_1.asc 51 46 26 31 64 61 51 48

tPhoto142_CV4_2.asc 38 27 44 23 60 38 42 52

tPhoto142_HV3_1.asc 33 59 32 17 49 67 48 55

tPhoto142_HV3_2.asc 36 109 43 19 57 118 42 54

tPhoto142_HV4_1.asc 70 52 42 28 91 66 43 50

tPhoto143_CV4_2.asc 43 30 43 22 64 41 42 53

tPhoto143_HV3_1.asc 30 59 31 15 45 66 48 56

tPhoto143_HV3_2.asc 39 102 43 21 60 112 42 53

tPhoto143_HV4_1.asc 62 53 43 28 83 67 42 50

tPhoto208_CV4_2.asc 43 49 43 29 64 63 42 49

tPhoto208_HV3_1.asc 48 50 34 29 65 64 47 49

tPhoto208_HV3_2.asc 54 72 32 25 70 84 48 51

tPhoto208_HV4_1.asc 44 55 43 17 65 63 42 55

tPhoto255_HV3_1.asc 44 53 38 21 63 63 45 53

tPhoto255_HV4_1.asc 41 49 46 26 64 62 41 51

tPhoto255_HV4_2.asc 42 51 46 22 65 62 41 53

tPhoto256_HV3_1.asc 47 54 37 16 65 62 45 56

tPhoto256_HV4_2.asc 41 51 46 23 64 62 41 52

tPhoto257_HV3_1.asc 46 53 37 18 64 62 45 55

tPhoto257_HV4_1.asc 40 51 46 23 63 62 41 52

tPhoto257_HV4_2.asc 43 49 46 26 66 62 41 51

tPhoto258_HV3_1.asc 49 56 37 15 67 63 45 56

tPhoto258_HV4_1.asc 43 50 48 29 67 64 40 49

tPhoto258_HV4_2.asc 42 51 48 23 66 62 40 52

tPhoto259_HV3_1.asc 50 55 36 16 68 63 46 56

tPhoto259_HV4_1.asc 43 48 48 31 67 63 40 48

tPhoto259_HV4_2.asc 41 50 47 27 64 63 40 50

tPhoto260_HV3_1.asc 50 53 35 16 67 61 46 56

tPhoto260_HV4_1.asc 38 48 48 34 62 65 40 47

tPhoto260_HV4_2.asc 36 45 45 37 58 63 41 45

tPhoto261_HV3_1.asc 48 75 35 17 65 83 46 55

tPhoto261_HV4_1.asc 43 46 48 35 67 63 40 46

tPhoto261_HV4_2.asc 37 51 48 22 61 62 40 53

tPhoto262_HV3_1.asc 47 73 36 17 65 81 46 55

tPhoto262_HV4_1.asc 44 55 48 20 68 65 40 54

tPhoto262_HV4_2.asc 43 51 48 29 67 65 40 49

tPhoto263_HV3_1.asc 43 78 35 17 60 86 46 55

tPhoto263_HV4_1.asc 48 53 48 20 72 63 40 54

tPhoto263_HV4_2.asc 41 55 48 21 65 65 40 53

tPhoto264_OV3_1.asc 47 51 36 23 65 62 46 52

tPhoto264_OV4_1.asc 51 54 48 18 75 63 40 55

tPhoto264_OV4_2.asc 44 53 48 19 68 62 40 54

tPhoto265_HV3_1.asc 48 46 32 30 64 61 48 49

tPhoto265_HV4_1.asc 52 52 47 20 75 62 40 54

tPhoto265_HV4_2.asc 42 52 47 21 65 62 40 53

tPhoto266_OV3_1.asc 45 45 35 28 62 59 46 50

tPhoto266_OV4_1.asc 62 53 32 22 78 64 48 53

tPhoto266_OV4_2.asc 39 48 47 25 62 60 40 51

tPhoto267_OV3_1.asc 48 49 34 29 65 63 47 49

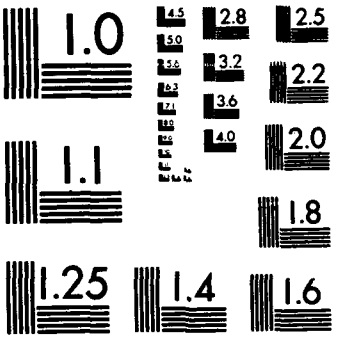
tPhoto267_OV4_1.asc 43 49 42 26 64 62 43 51

tPhoto267_OV4_2.asc 41 53 46 23 64 64 41 52

tPhoto268_HV3_1.asc 46 50 36 23 64 61 46 52

tPhoto268_HV4_1.asc 42 45 44 35 64 62 42 46

AD-A259 003 CORRELATION BASED TARGET LOCATION AND IDENTIFICATION 2/2
(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
SCHOOL OF ENGINEERING R A WENZEL DEC 92
UNCLASSIFIED AFIT/GSO/ENG/92D-05 XC-AFIT NL



RESOLUTION TEST CHART

tPhoto268_HV4_2.asc 43 54 46 18 66 63 41 55

tPhoto269_HV4_1.asc 45 47 44 34 67 64 42 47

tPhoto269_HV4_2.asc 46 53 46 17 69 61 41 55

tPhoto269_OV3_1.asc 47 56 35 17 64 64 46 55

Appendix F. *KHOROS*

KHOROS is an image processing package which is basically a set of approximately 260 routines. Cantata is the visual programming environment for the KHOROS system. To execute cantata, simply enter "cantata" on the command line in the cmdtool on a SUN workstation. It will take about one minute for cantata to completely load and display the masterform workspace on the screen.

F.1 Using KHOROS

Designing a sequence of KHOROS commands can be done in two ways. First, if the exact command is known, it can be located using the "ROUTINES" box on the upper left side of the masterform. Clicking on "ROUTINES" using the left mouse button will bring up a list of all available routines. Use the left mouse button to click on all routines needed. Once this is done, select "Use" using the left mouse button. The list will disappear and faint outlines of the routine(s) selected will appear under the cursor. Place the cursor in the masterform where you want the glyphs and click the left mouse button. The glyphs (icons) will appear on the masterform.

The second way to choose glyphs is to use the pull-down menus located at the top of the masterform. Select the correct menu by holding down the right mouse button and pulling the menu down. When the correct routine is highlighted, release the right mouse button. The faint outline of the selected routine will appear under the cursor. Place the cursor in the masterform where you want the glyphs and click the left mouse button.

Any glyph selected that is no longer needed can be deleted by selecting the "bomb" image in the upper left corner of the glyph using the left mouse button. The center blue image next to the "bomb" image is used to change default values for the glyph. The upper right image on the glyph resembles a light switch. Clicking this with the left mouse button will execute that glyph.

The glyphs also have boxes along their left and right edges, each with an arrow that points to the right. The arrows on the left edge are "input" arrows and the arrows on the right edge are output arrows. To connect the output of one routine to the input of another, simply click on the output arrow of the first routine and then click on the input arrow of the second routine using the left mouse button. The connection will now show on the screen. To delete a connection, place the cursor on the it and click the left mouse button. This will bring box which has "Delete Connection" on the upper left. Clicking on this with the left mouse button will delete the connection. However, sometimes the connection will not be erased from the screen. This can be rectified using the "REDRAW" command as discussed below.

Once the appropriate routines are on the masterform, they can be executed using the "RUN" box at the upper left of the master form. Each routine will be executed in order and the output of one routine will automatically be fed to the input of the next routine following the routes you have designated. If any routine does not execute correctly, a blinking, red, unhappy face will appear on the glyph. This can be selected to view an appropriate error message.

Other boxes located on the upper left of the master form include "REDRAW" which will clean up the masterform and erase any connections or glyphs that have been deleted. "RESET" will reset all glyphs if any changes have been made to them. "HELP" will give some information about KHOROS and on-line documentation available. "QUIT" will quit KHOROS.

F.2 Correlation using KHOROS

To generate a basic correlation sequence, you will need perform the following routines. Each routine is discussed along with where I located it

F.2.1 asc2viff Asc2viff converts the raw data (ascii) to the VIFF format needed. It is found under "CONVERSIONS" and "Raw File Format'.

You can designate the path where your ascii file resides, or else click on "Input ASCII file" using the left mouse button. This will display your home directory. The path to the correct input file can be designated by sequentially selecting the appropriate directories/subdirectories using the left mouse button. Other important boxes are as follows. Make sure that the "ASCII to VIFF" box on the upper left is highlighted in black. Also, the "data type" is "REAL". Finally, fill in the number of rows and columns with the size of your image. Once all is correctly filled in, select "Glyph" in the upper left corner.

F.2.2 vextract Vextract will extract a set of pixels. It is found under "IMAGE PROCESSING" and "Subregion", and "Extract Subimage". To designate the correct set of pixels, input the correct "Upper left X coord.", "Upper left Y coord.", "Width in Pixels", and "Height in Pixels". Select "Glyph".

F.2.3 vpad Vpad will take the extracted image and pad around it with a designated pixel value. It is found along with vextract, except you select "Pad Image" instead of "Extract Subimage". Designate the size of the padded image in "Row size" and "Column size". You can control where the extracted image is placed in the padded image by designating the "Offset of row" and "offset of column". Finally, designate the Real and Imaginary "Pad Value". For my purposes, both "Pad Values" were 0.

F.2.4 vfft Vfft will find the fast Fourier transform of an image. It is found under "IMAGE PROCESSING", "Transforms", and "FFT". Specify the FFT direction as "Forward" or "Inverse" by using the left mouse button on the "Forward" or "Inverse" box, i.e. this box toggles.

F.2.5 vconj Vconj will find the complex conjugate of an image. It is found under "ARITHMETIC", "Unary Arithmetic", and "Conjugate".

F.2.6 vmul Vmul will multiply two images pixel by pixel. It is found under "ARITHMETIC", "Binary Arithmetic", and "Multiplication".

F.2.7 vtranslat Vtranslat will translate an image. It is found under "IMAGE PROCESSING", "Geometric Manip.", and "Translate Image". Make sure that the "X_Translation" and "Y_Translation" are exactly half of the width and height, respectively, of the size of the images you are correlating. This relocated the position of the DC component found by the Fourier transformations.

F.2.8 vconvert Vconvert will perform Viff file to other files. It is found under "CONVERSIONS" and "Data Conversions", and is used to convert Viff to byte formats for collecting statistics.

F.2.9 vpostscr Vpostscr will Print a hardcopy of an image when used in conjuction with Vconvert. It is found under "OUTPUT" and "Print Image". It accepts byte file format from the vconvert routine and sends to the designated printer.

F.2.10 put_update Put_update will display an image to the screen. It is found under "OUTPUT" and "Display Images".

F.2.11 vstats Vstats will compute statistics on a Viff file. It can be found under "ROUTINES". Feeding this to a xvviewer will show a two dimension plot of the correlation values.

F.2.12 xvviewer Xvviewer will display to the screen a file. When used in conjunction with vstats is will list various statistics for you to read.

F.2.13 xprism3 Xprism3 will display a plot to the screen. When used in conjunction with vstats it will display a three dimension plot of the correlation.

F.3 Invoking KHOROS using C and C Shells

While the KHOROS Cantata command form is a very useful tool, it is very time consuming to use. One way around this is to invoke KHOROS from within a C program. Since each KHOROS routine is itself executable via a system level command, this is easily done. The following is a step by step procedure to do this:

- Build the KHOROS network using Cantata. Determine which routines are necessary and in what order. Also note which parameters in the KHOROS routines are image dependent, that is, which paramaters change as the image changes. These will be passed as program parameters in the C program.
- Debug the network. This is self explanatory. The network must work as you intend it to work.
- Reset the Cantata masterform. This will ensure all routines are executed one last time for the "capture".
- Capture the KHOROS execution. Once the network has successfully executed, quit the KHOROS masterform. In the command tool that was used to invoke KHOROS, the execution of the network is documented. Simply highlight the various routine executions with the left mouse button, hit "copy" on the left side of the keyboard, and place in a new file by hitting "paste". These will be made into a C Shell.
- Build the C Shell. In the file that contains the KHOROS execution capture, place the following as the first line

```
#!/bin/csh
```

. Next, determine which variables will be passed to the C Shell and in which order. For example, if the C Shell is called *correlate* and the image filename, x, y, w, and h values need to be passed, then the shell can be called by *correlate filename x y w h* with the appropriate values in the corresponding place. Inside the shell, these values can be referenced by $1, $2, $3, $4, and $5 respectively. Determine which of the KHOROS routines have variable parameters and replace the numeric values in the capture with the $ variable name. Or symbolic names can be used by using the "set" command. For example, "set infile = $1" will allow the input image file to be referenced by "$infile" for ease of use. If the output file is to have the same root name as the input image, but with a different extension, then use a two line sequen

- Make the C Shell executable. Type *chmod 766 correlate* for this example.
- Build the input data file. The data file is simply a laundry list of input file name and all necessary variable parameters to be used by the C Shell.
- Build the C program. The C program will open the input data file for reading via the fscanf function. Next, a "for" loop can be used to have the program execute for the correct number of files listed in the input data file. A fscanf function is used to read in the image to be processed, along with the associated variable parameters. The actual command to invoke the C Shell with the correct parameters must now be placed into a string. This is done using a sprintf command, such as the following example: "sprintf(cmd, "correlate %s %d %d %d %d", image, x,y,w,h);" will place the call to the C Shell in the string "cmd". This can now be invoked by the line "system(cmd);".
- Compile the C program. Self explanatory.
- Run the C code. Self explanatory.

An example of this can be found in Appendices D.5 and D.6 under C programs and C shells. *Correlate* is the C Shell that contains the KHOROS capture. The C

program *docorrelate.c* reads in data file autocor.dat, and passes these parameters to the *correlate* C Shell 180 times. The output was 180 files with the .dat extension.

Appendix G. *ANVIL*

ANVIL, the Artificial Neural Vision Learning System, was developed by Booz-Allen & Hamilton, Inc. of Arlington, Virginia under Contract Number F33615-89-C-1013 for Wright Laboratories AAAT-1 at Wright-Patterson AFB, OH. *ANVIL* is an autonomous target recognition (ATR) software package that takes a training set of HIPS formatted terrain board imagery, learns features for targets taken from these images, and develops a neural network to locate these targets. The description of how *ANVIL* works follows.

G.1 How ANVIL Works

ANVIL can conceptually be broken up into three stages (2):

- Training – finding features in a training set that can be used to distinguish between training images
- Associative Mapping – associating the feature positions for each training image with the appropriate image
- Performance – looks for the same set of feature in the same geometry in the test image

A discussion of each stage follows.

G.1.1 Training *ANVIL* uses a type of template matching approach for the ATR process. Each training image is segmented by hand to identify the target from the background. Once an image is segmented, *ANVIL* finds a set of templates, called *features*, that are 11 x 11 pixels in dimension. The following discussion describes how the features are found.

The first training image has the segmented target extracted from the image and padded with low intensity values to make up the original 128 x 128 image size.

This image is then compared with an 11 x 11 pixel mask called the *novel mask*. This mask has a high intensity pixel in its center and then each concentric square around this has decreasing intensity – a type of gaussian mask. The mask is applied to every possible place in the 128 x 128 training image and the similarity metric MSE calculated. This results in a 118 x 118 matrix of MSE calculations called a *novel Cluster*. If any 11 x 11 set of pixels in the input image has all pixel values less than 20, the corresponding error value in the novel cluster is forced to the maximum error value of 1.0. This step, called *Information Content Gating* is designed to remove low intensity areas with little or no information content. Next, any pixel in the 118 x 118 novel cluster that is above the *information threshold* will be replaced with the maximum error value of 1.0. Now, we find the minimum MSE in the 118 x 118 novel cluster. Once the minimum error term is found, we find the corresponding 11 x 11 feature in the input image. This becomes the first learned feature.

This learned feature now becomes an influence into how the novel mask learns the next feature, whether this new feature comes from the same input image or from a successive input images.

The second input image is presented for training. We now have an 11 x 11 training feature as well as the 11 x 11 novel mask. Again, the input image is presented to the novel mask and a 118 x 118 novel cluster results. Again, any 11 x 11 region in the input image that has all 121 pixels below the threshold value causes the corresponding value in the novel mask 118 x 118 error matrix to be set to 1.0.

The second input image is also presented to the first 11 x 11 learned feature which also results in an 118 x 118 cluster called a *feature extraction cluster* (FE cluster). Since we have only the one learned feature at this time, this FE cluster has no other FE cluster to compare with, so it is directly gated to the *Learned Cluster Column Exclusivity*. If any error term in the FE cluster is below a threshold value, called the *separation threshold*, the corresponding term in the *novel cluster*, not the FE cluster, is set to maximum error of 1.0. The FE cluster is now sent through *Fea-*

ture Vigilance Learned Gating. Any error term above the *feature vigilance threshold* in the FE cluster will be forced to the maximum error of 1.0. The minimum error in the FE cluster is found and the values within the *exclusivity region* surrounding this minimum error are forced to the maximum error of 1.0. This process is repeated until there is only one error value in the FE cluster for each exclusivity region not set to the maximum error value. The surviving FE cluster error values force the corresponding novel cluster error values to the maximum error value. That is, if there exists a low error value in the FE cluster, then this feature must already be represented and needs to be excluded from being learned again. The novel cluster is searched for the minimum error value. Once this is found, the corresponding 11 x 11 pixel are in the input image is extracted and becomes the new learned feature.

The processing now begins anew with either a new training image, or by cycling through the training images again. Again, the training image is compared with the novel mask, but it is also compared with the two learned features. This results in two FE clusters. These two FE clusters are compared on a "pixel-by- pixel" basis and the lowest error for each "pixel" is maintained in an overall *combined learned cluster.* This process is called the Learned Cluster Column Exclusivity and is performed when there are two or more learned features.

The processing as mentioned above is continued until the novel mask is composed of only maximum error values. At this time, feature learning is completed and the next step is begun.

G.1.2 Associative Mapping Once all of the training images are segmented and the features are learned, each image is processed again. This stage records the various features and their relative positions in each training image. The locations of the features with respect to the center of the segmented target are used to set up the *Spatial Location Layer masks.* The learned features are also combined into a content addressable memory (CAM), which is the region around a training image where it is

chosen as the best above other training images. From the features and their relative positions, *ANVIL* then builds a neural network architecture based on this to find targets in the test images. While *ANVIL* is neither perspective or scale invariant, it is *robust* to scale and perspective changes. Booz–Allen & Hamilton claims the amount of robustness to be ± 10 degrees azimuth, ± 5 degrees elevation and ± 25 percent in scale.

G.1.3 Performance Test images are now fed to the newly learned neural network architectur . In the feature extraction layer, *ANVIL* compares each training feature with *every possible position* in the input image and computes a similarity metric which measures how closely the feature matches that image at that position. These similarity metrics are then combined into an array of outputs, called *clusters*, for each learned feature.

The spatial location layer looks for the best feature match within the corresponding mask and computes a similarity metric. Again, this is performed for all possible positions, and results in arrays that record the positions of the best feature matches *relative to the spatial locations mask's center.*

Finally, in the object detection layer, feature and position information are combined into an overall target similarity metric, resulting in a similarity measure for every training training target at every possible target location. This similarity measure is the overall error between the image and the training target at that location. The lowest values become candidates for target detections. However, a threshold value is also used to eliminate those with higher values.

G.2 ANVIL System Requirements

According to the *ANVIL Software User's Manual*, the *ANVIL* system runs ONLY on the following list of hardware:

- Sun's Operating System (SunOS) 4.0.3 or higher

- Color Sun-4 and SPARC workstations with a three button mouse
- Both 8 bit and 24 bit color (CG9) is supported
- Inmos transputers
- 20 MB of disk space for the source code

Bibliography

1. Ballard, Dana H. and Christopher M. Brown. *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1982.
2. Booz-Allen & Hamilton, Inc. *ANVIL Interim 1990-1991 Interim Progress Report*, 6 December 1991. Contract F33615-89-C-1013. Wright Laboratories AAAT-1. Wright-Patterson AFB, OH.
3. Booz-Allen & Hamilton, Inc. *Artificial Neural Vision Learning System (ANVIL) Software User's Manual.* Contract F33615-89-C-1013. Wright Laboratories AAAT-1. Wright-Patterson AFB, OH.
4. Casasent, David and Demetri Psaltis. "Position, Rotation, and Scale Invariant Optical Correlation," *Applied Optics, 15:* 1795-1799 (July 1976).
5. Cerella, John. "Pigeons and Perceptrons," *Pattern Recognition, 19:* 432-437 (June 1986).
6. Cline, John D. *Hybrid Optical/Digital Architecture for Distortion Invariant Pattern Recognition.* MS thesis, AFIT/GEO/ENG/89D-2. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1989.
7. Daugman, John G. "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing, 36*(7):1169-1179 (July 1988).
8. Gaskill, Jack D. *Linear Systems, Fourier Transforms, and Optics.* John Wiley & Sons, 1978.
9. Gonzalez, Rafael C. and Paul Wintz. *Digital Image Processing* (Second Edition). Reading, MA: Addison-Wesley Publishing Co., 1987.
10. Goodman, Joseph W. *Introduction to Fourier Optics.* San Francisco: McGraw-Hill Book Co., 1968.
11. Hazlett, Michael A. *Gabor Segmentation of High Resolution Synthetic Aperature Radar Imagery.* MS thesis, AFIT/GE/ENG91D-23. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.
12. Law, Paul D. *Correlation Based Distortion Invariant Infrared Tracking.* MS thesis, AFIT/GE/ENG/91D-35. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.
13. Mayo, Michael W. *Computer Generated Hologram and Magneto-Optic Spatial Light Modulator for Optical Pattern Recognition.* MS thesis, AFIT/GEO/ENG/87D-1. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

14. Rasure, John *et al.*, KHOROS Manuals, The Vision Lab, Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, 1991.

15. Rogers, Steven K. *et al. An Introduction to Biological and Artificial Neural Networks.* Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, October 1990.

16. Roggemann, Michael C. *Multiple Sensor Fusion for Detecting Targets in FLIR and Range Images.* PhD dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, May 1989.

17. Ruck, Dennis W. *Multisensor Target Detection and Classification.* MS thesis, AFIT/GE/ENG/87D-56. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

18. Smiley, Steven E. *Image Segmentation Using Affine Wavelets.* Ms thesis, AFIT/EN/ENG/91D-50. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1991.

19. Tong, Carl W. *Target Segmentation and Image Enhancement through Multisensor Data Fusion.* MS thesis, AFIT/GE/ENG/86D-55. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986.

20. Troxel, Steven E. *Position, Scale, and Rotation Invariant Target Recognition Using Range Imagery.* MS thesis, AFIT/GEO/ENG/87D-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1987.

21. Walrond, J. Thomas and Timothy G. Childress *Position, Scale, and Rotation Invariant Optical Pattern Recognition for Target Extraction and Identification.* MS thesis, AFIT/GE/ENG/88D-4. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1986.

Vita

Richard A. Wenzel was born 28 February, 1962, in New Braunfels, Texas. He graduated from Navarro High School, in Geronimo, Texas, in 1980. He attended Southwest Texas State University, receiving his BS in Computer Science in 1984. After marrying his wonderful wife, Terri, he attended Officer Training School and received his commission in April 1985. He served at Onizuka AFB, California as an Assistant Project Officer with the Satellite Operations Training Program from 1985 to 1987. He then transferred to Peterson AFB, Colorado, where he graduated from Satellite Operations Training Program as a Mission Controller and Deputy Flight Commander with the Defense Satellite Communications Program III (DSCS III). He also served as a simulation developer, rehearsal engineer, and DOT instructor. He entered the Space Operations Masters Program in the School of Engineering, Air Force Institute of Technology, in June 1991. Upon graduation, he will separate from the Air Force.

Permanent address: Rt. 7 Box 322
Seguin, Texas 78155

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
CORRELATION BASED TARGET LOCATION AND IDENTIFICATION

5. FUNDING NUMBERS

6. AUTHOR(S)
Richard A. Wenzel
Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSO/ENG/92D-05

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Correlation was investigated as a technique for locating and identifying visible light terrain board images of tanks, airplanes, and trucks. A correlator was built which utilizes the Fast Fourier Transform. The 180 images in the image set were correlated in both the raw form and after pre-processing via energy normalization and local energy normalization. The correlation of the raw images led to development of quantitative and heuristic rules for possible inclusion in a rule based expert system. The correlation of the local energy normalized images resulted in a 100 percent target location rate and can be used as a baseline model based vision system with which other techniques can compare.

14. SUBJECT TERMS
Computer Vision, Correlation, Fast Fourier Transform, Image Processing

15. NUMBER OF PAGES
112

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL